Niveau(x) d'enseignement: 7 Numéro EDUQ : 8170539
Centre d'intérêt: 92

Bard, Pierre.

FICTIF-2 : un ordinateur simplifié et sa programmation / par Pierre Bard . -- Rimouski : Collège de Rimouski, 1974.
114 f. ; 28 cm.

Cote Dewey: 001.642/B245f Cote LC: QA/76.6

Descripteur(s): *Enseignement par ordinateur *Ordinateur *Informatique *Simulation *Programmation *Manuel scolaire *Collégial (Niveau)

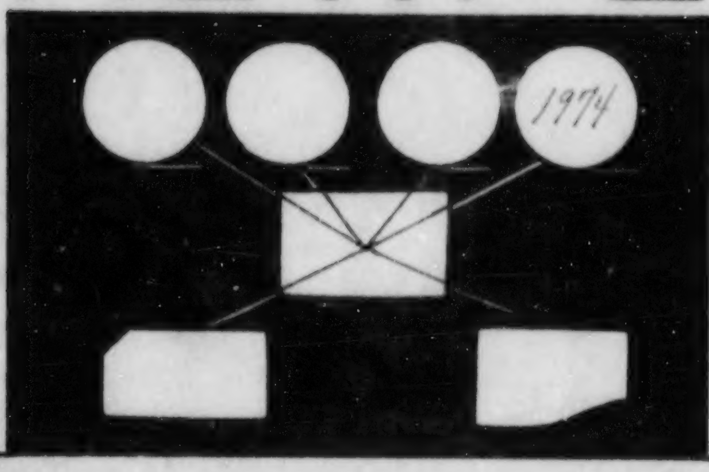
Identificateur(s) de nom: *FICTIF-2 (Ordinateur)

Manuel scolaire conçu dans le cadre d'un projet d'enseignement assisté par ordinateur réalisé au Cégep de Rimouski. Le projet, qui concerne les cours d'informatique, consiste en la simulation d'un ordinateur simplifié à partir d'un ordinateur en soi trop complexe pour le niveau de connaissance des étudiants. Outre le programme de simulation, l'auteur a rédigé ce texte qui doit guider l'élève dans l'utilisation de l'ordinateur. Cet outil doit lui permettre d'apprendre vite et bien la manière dont fonctionne un ordinateur (étude du matériel), la nature du travail qu'il peut effectuer (étude de la programmation) et finalement comment, dans la pratique, l'ordinateur est secondé dans son travail par des programmes spéciaux (logiciel) et par du personnel (opérateur).

Type de document: d
Financement: Bénéficiaire - Collège de Rimouski
Organisme(s) source(s) - MEQ - DGEC
Programme(s) de subvention - PROSIP
Auteur de l'analyse: ar

EDUQ 8170539

FICTIF-2



UN ORDINATEUR SIMPLIFIÉ ET SA PROGRAMMATION

PIERRE BARD

FEB 1975
REÇU
DIGEC

E D U Q 8170539

F I C T I F - 2

Un ordinateur simplifié
et sa programmation

par Pierre BARD
professeur d'informatique
au Collège de Rimouski

Rimouski
Collège d'Enseignement général et professionnel
1974

Table des matières.

Pages

Introduction 6

PARTIE 1.

Chapitre 1.

L'ordinateur FICTIF-2. Présentation sommaire. 9

1.1. Les différentes parties de l'ordinateur. 9

1.2. Schéma fonctionnel du FICTIF-2. 12

Chapitre 2.

La mémoire centrale du FICTIF-2. Notions de base. . . . 13

2.1. Définitions. 13

2.2. Structure logique d'une mémoire et principes de base. . . 15

2.3. La mémoire du FICTIF-2. 16

2.4. Image-mémoire ou "DUMP". 18

2.5. Instructions. 19

Chapitre 3.

Eléments de programmation en langage machine. 21

3.1. Codes d'opération (en partie). 21

3.2. Un programme simple. 22

3.3. Codes d'opération (en partie). 23

3.4. Un autre programme simple. 25

Chapitre 4.

Les registres d'index et l'incrémentation automatique. . 27

4.1. Rôle des registres d'index. 27

4.2. Exemple d'utilisation de registre d'index. 28

4.3. Règles d'utilisation des registres d'index. 30

4.4. Incrémentation automatique. 30

4.5. Exemple d'utilisation de l'incrémentation automatique. . 31

Chapitre 5.

Les périphériques de l'ordinateur FICTIF-2. 33

5.1. Le lecteur de cartes. 33

5.2. L'imprimante. 34

5.3. Les dérouleurs de bandes magnétiques. 35

Chapitre 6. Pages

Le langage machine du FICTIF-2. Compléments. 39

6.1. L'instruction d'édition. 39

6.2. Saut avec adresse de retour. 41

6.3. Précisions sur l'addition et la soustraction. 42

6.4. Saut sur dépassement. 45

6.5. Compléments d'information sur les opérations arithmétiques et de déplacement. 45

6.6. Tableau récapitulatif des codes d'opération. 46

Chapitre 7.

L'unité de contrôle. 47

7.1. Format des instructions. 47

7.2. Fonctionnement. 49

7.3. Instructions câblées et non câblées. 53

Chapitre 8.

L'unité arithmétique et logique. 55

Chapitre 9.

Programmation en LMF. 57

Chapitre 10.

Programmation en SYMBOLIC. 63

Chapitre 11.

Eléments de système d'exploitation. 77

11.1 Le moniteur d'enchaînement. 78

11.2 Le chargeur LMF. 82

11.3 Le compilateur (assembleur) SYMBOLIC. 83

11.4 Le générateur d'image-mémoire. 84

PARTIE II.

Chapitre 12.

Le superviseur. 87

12.1 Généralités. 87

12.2 Constitution du superviseur. 89

12.3 Fonctionnement. 90

Pages

12.4 Réflexions au sujet du superviseur. 91

12.5 Emplacement des registres de l'unité de contrôle. . . . 92

Chapitre 13.

La console (ou pupitre de commande). 93

13.1 Aspects généraux. 93

13.2 Précisions sur l'amorce. 95

13.3 Exemple d'interaction entre le superviseur, la console et l'opérateur. 95

Chapitre 14.

Contrôle interne de l'ordinateur. 97

14.1 Trappe PE (erreur de programme). 97

14.2 Trappe ME (erreur de mémoire). 98

14.3 Trappe HI (horloge interne). 99

ANNEXE 1.

Catalogue des messages et erreurs. 101

ANNEXE 2.

Tableau des codes d'opération. 114

Introduction.

Il n'est pas toujours simple d'initier correctement une personne à une matière donnée. La première erreur consiste à simplifier les choses à l'extrême et à ne présenter des faits qu'une fraction tellement peu significative que l'initié devra ultérieurement poursuivre son exploration longtemps avant d'avoir une perception correcte de la réalité. La deuxième erreur, lorsque la matière est complexe, est de l'exposer avec un tel souci du détail que l'essentiel n'est pas dégagé.

En informatique, le problème se pose avec acuité. En effet, l'étude des ordinateurs devient de plus en plus difficile, compte tenu du fait que les constructeurs déploient des efforts inouis d'imagination pour rendre leurs machines de plus en plus adaptées aux besoins des usagers. Si plusieurs des concepts sous-jacents à la construction des ordinateurs actuels dépassent de beaucoup le niveau du débutant, il n'en reste pas moins que, dans ses principes généraux, l'ordinateur n'a pas changé. Mais un travail d'abstraction est nécessaire.

Nous avons ressenti le besoin de donner à l'étudiant un outil d'apprentissage qui lui permette d'apprendre vite et bien la manière dont fonctionne un ordinateur (étude du matériel), la nature du travail qu'il peut effectuer (étude de la programmation) et finalement comment, dans la pratique, l'ordinateur est secondé dans son travail par des programmes spéciaux (logiciel) et par du personnel (opérateur). Nous avons ainsi le sentiment de ne pas négliger d'éléments importants de l'informatique tout en présentant, dans chaque cas, des concepts simplifiés, il est vrai, mais débouchant sur la réalité.

Aussi, dans le cadre d'un programme d'aide à la recherche subventionné par la DIGEC, un organisme du Ministère de l'Education du Québec, nous avons imaginé l'ordinateur FICTIF-2. Le coeur du projet réside dans la simulation de cet ordinateur fictif sur un ordinateur véritable, le CENTURY-100 de NCR. Nous avons donc programmé ce dernier pour "prêter" une

partie de ses capacités à notre ordinateur FICTIF-2. Il ne faudra pas s'étonner que celui-ci apparaisse comme un CENTURY-100 simplifié et vulgarisé.

Comme résultat, l'étudiant a l'avantage de pouvoir faire des travaux concrets de programmation. Le présent texte se veut un manuel pour le guider dans l'utilisation de cet ordinateur et pour lui faire comprendre, autant qu'il est possible à ce niveau, les aspects de l'informatique que nous avons nommés plus haut.

Le texte est divisé en deux parties. La première correspond aux notions que le débutant doit acquérir dans un cours d'un semestre et qui doivent être complétées par des travaux pratiques suggérés par un professeur. La deuxième partie, beaucoup plus courte, touche des points qui habituellement, nous le regrettons, ne sont pas abordés à ce niveau. Pourtant, ceux qui s'adonneront à l'étude de ces points verront s'améliorer leur compréhension de l'ordinateur. L'initiation sera complétée.

PARTIE I.

Chapitre 1.

L'ordinateur FICTIF-2. Présentation sommaire.

Qu'une machine mérite le nom d' ordinateur, cela tient moins à la rapidité avec laquelle elle peut effectuer certaines opérations qu'à la présence d'éléments bien définis (mémoire centrale, unité de contrôle, unité arithmétique et logique, unités d'entrée-sortie) et qu'à une organisation déterminée de ces éléments entre eux.

1.1 Les différentes parties de l'ordinateur.

A- L'unité de contrôle.

Pour que l'ordinateur travaille, il est primordial qu'il obéisse à un programme, qui est une suite d'instructions. Ce programme est emmagasiné, pour la durée de son exécution, dans la mémoire centrale. L'unité de contrôle a comme rôle d'extraire une à une les instructions de la mémoire et de veiller à leur exécution. Par exemple, si l'unité de contrôle rencontre une instruction de lecture d'une carte perforée, elle donnera ordre d'une part au lecteur de cartes de faire passer la première carte disponible devant des cellules photoélectriques qui libéreront un train d'impulsions correspondant à la présence et à l'absence de perforations et d'autre part à la mémoire d'emmagasiner l'information portée par ce train d'impulsions.

L'ordinateur, on s'en doute, est incapable de prendre des initiatives. Sa fiabilité est d'ailleurs liée à ce fait. Il demeure l'exécutant fidèle des ordres qu'on lui donne; si ceux-ci sont incorrects, les résultats seront également incorrects. Tout cela revient au fait suivant: l'unité de contrôle constitue le centre nerveux de l'ordinateur et aucune autre partie ne peut faire quoi que ce soit sans en avoir préalablement reçu l'ordre de cette unité. Par ailleurs, l'unité de contrôle suit fidèlement, telle une partition musicale, les instructions préparées à l'avance par un programmeur et entreposées dans la mémoire centrale. La responsabilité de la bonne exécution d'un travail par l'ordinateur incombe donc au programmeur.

B- La mémoire centrale.

La mémoire centrale doit conserver les instructions du programme en cours d'exécution, recevoir les informations des organes d'entrée comme le lecteur de cartes, fournir les résultats aux organes de sortie comme l'imprimante. Si des opérations arithmétiques doivent être exécutées par l'unité arithmétique et logique, la mémoire centrale fournit les données et reçoit les résultats.

Si la mémoire humaine est une faculté constamment ouverte sur le monde, celle de l'ordinateur n'enregistre pas ce qu'elle veut. Au contraire, ce n'est que lorsque l'unité de contrôle lui en donne l'ordre qu'elle peut entrer en action pour recevoir ou transmettre de l'information.

La mémoire centrale de l'ordinateur a une capacité généralement limitée et il est fréquent qu'on détruise des informations devenues inutiles pour les remplacer par des nouvelles. Il est alors important de connaître exactement la position de chacune d'elles.

C- L'unité d'entrée.

Dans le cas de l'ordinateur FICTIF-2, l'unité d'entrée est un lecteur de cartes. On retrouve cette unité sur la plupart des ordinateurs actuels mais il existe une variété de plus en plus grande de moyens de faire entrer de l'information dans l'ordinateur.

Un lecteur de cartes possède un magasin d'alimentation dans lequel on peut placer plusieurs centaines de cartes en attente de lecture, un mécanisme de lecture qui extrait la première carte disponible du magasin d'alimentation et la fait passer devant des cellules photoélectriques et finalement un magasin de réception qui empile une à une les cartes lues. Ce dernier peut également contenir plusieurs centaines de cartes. La vitesse de lecture se situe entre 100 et 2000 cartes à la minute selon la puissance du lecteur et de l'ordinateur.

D- L'unité de sortie.

Un ordinateur serait inutile, dans la plupart des cas, s'il ne lui était pas possible de transmettre, sous forme lisible, les résultats d'un travail. C'est l'imprimante qui effectue cette opération. L'imprimante, comme la mémoire centrale et l'unité d'entrée, est assujettie à l'unité de contrôle. L'information imprimée origine directement de la mémoire centrale.

Une imprimante classique possède un débit de quelques centaines de lignes à la minute. Certaines imprimantes rapides peuvent toutefois soutenir un rythme de quelques milliers de lignes à la minute.

E- L'unité arithmétique et logique.

Le rôle de cette unité est évidemment d'effectuer les opérations classiques d'addition, de soustraction, de multiplication et de division. Les données nécessaires aux opérations viennent de la mémoire et les résultats retournent à la mémoire. Il est donc impossible de passer directement de cette unité à l'imprimante: toute information transite dans la mémoire comme le montrera le schéma à la section 1.2.

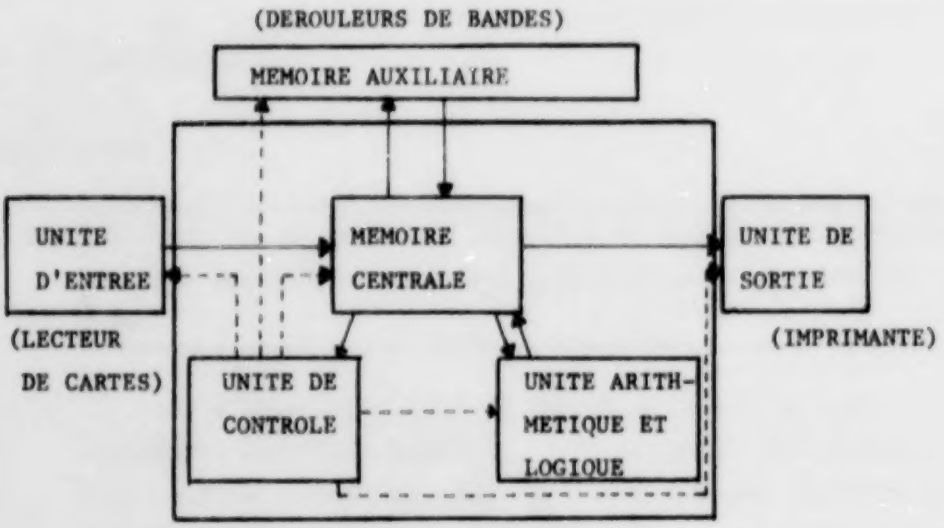
Quant aux opérations logiques, elles consistent ici en la comparaison de deux informations en mémoire pour déterminer si elles sont semblables ou différentes. Lorsqu'un programmeur écrit les instructions pour un travail donné, il utilise souvent la comparaison pour faire exécuter telle séquence d'instructions plutôt que telle autre. Il en résulte une grande souplesse dans le traitement des informations dès lors qu'il est possible, par exemple, de prendre un chemin différent en traitant l'information concernant un homme ou une femme, une personne mariée ou célibataire, etc.

F- La mémoire auxiliaire.

Parce qu'il est souvent nécessaire de conserver l'information longtemps et en grande quantité, on a recours à un support d'information de type magnétique qui a le mérite d'être réutilisable. L'ordinateur FICTIF-2 est pourvu d'un groupe de dérouleurs de bandes magnétiques. Un dérouleur est unité d'entrée si on lit l'information sur la bande ou unité de sortie si on y réenregistre de nouvelles informations.

Plusieurs sortes de mémoires auxiliaires ont été inventées. Les disques magnétiques constituent une solution de rechange fort répandue aujourd'hui par rapport aux rubans magnétiques. Ces derniers ont cependant été retenus ici à cause de leur simplicité et parce qu'ils suffisent amplement aux besoins de ce cours.

1.2 Schéma fonctionnel du FICTIF-2.



Lignes pointillées: elles indiquent que, pour entrer en action, les unités recevant une telle ligne doivent en obtenir l'ordre de l'unité de contrôle.

Lignes pleines: elles indiquent un transport d'information dans le sens donné. On voit aisément sur cette figure que toute information passe par la mémoire centrale. En particulier, on remarquera la ligne pleine allant de la mémoire à l'unité de contrôle: elle représente le passage de ce type particulier d'informations que sont les instructions du programme situées, comme nous l'avons mentionné plus haut, dans la mémoire centrale.

Chapitre 2.

La mémoire centrale du FICTIF-2. Notions de base.

2.1. Définitions.

2.1.1. Le "bit".

Déf: Le bit est l'unité élémentaire d'information et sa valeur est soit 0, soit 1.

Bit est une contraction de "BInary digiT" qui signifie "chiffre binaire".

2.1.2. L'octet.

Déf: Un octet est un groupe de huit bits et représente normalement un chiffre décimal, un caractère alphabétique ou un caractère spécial.

Les différents groupes de huit bits possibles sont: 00000000, 00000001, 00000010, ..., 11111111. Cela donne en tout 256 configurations possibles. Cela est plus que suffisant pour représenter les quelques cinquante symboles que forme l'ensemble des lettres, des chiffres et des caractères spéciaux.

2.1.3. Un code.

Déf: Un code est une table des équivalences faites entre les configurations choisies et les symboles (lettres, chiffres, caractères spéciaux) qu'elles représentent.

Ainsi pour la mémoire du FICTIF-2, un code a été choisi. C'est un sous-ensemble du code international ASCII-8. Il existe d'autres codes internationaux, en particulier le code EBCDIC utilisé par la compagnie IBM.

Dans le code ASCII-8,

0100 0001 a été choisi pour représenter A,
0100 0010 " " " B,
0100 0011 " " " C,
. .
. .
. .
0011 0000 " " " 0,
0011 0001 " " " 1,
. .
. .
. .
0011 1001 " " " 9.

Voici sous forme de tableau le code choisi pour FICTIF-2:

CODE INTERNE DU FICTIF-2

b_4-b_1	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0010	SP			£	$			'	(	)	*	+	,	-	.	/
0011	0	1	2	3	4	5	6	7	8	9	:		<	=	>	
0100		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	P	Q	R	S	T	U	V	W	X	Y	Z					

b_8-b_5

SP = espace. Les carrés vides correspondent à des configurations inutilisées.

En abscisse, on retrouve les 4 bits de droite et en ordonnée, les 4 bits de gauche.

2.2. Structure logique d'une mémoire et principes de base.

Un très grand nombre d'ordinateurs utilisent l'octet comme élément logique de mémoire. On évalue alors la capacité d'une mémoire au nombre d'octets qu'elle peut contenir. Les plus petits ordinateurs ont une mémoire de moins de 10,000 octets tandis que les plus gros ont une capacité de quelques millions d'octets.

Pour emmagasiner en mémoire l'information suivante,

E	L	E	M	E	N	T	S		D	'	I	N	F	O	R	M	A	T	I	Q	U	E

23 octets seront nécessaires car il faut considérer que l'espace et l'apostrophe sont des caractères comme les autres. Il est évident que ces caractères doivent se succéder sinon leur repérage poserait un grave problème. Mais quel sens prend ici le mot "succession"? Dans une mémoire à octets, chaque octet possède un numéro d'ordre ou "adresse". Pour une mémoire de 10,000 octets, l'adresse du premier octet est 0000 et celle du dernier, 9999. On compare souvent une mémoire d'ordinateur à un ensemble de cases postales dont chacune a un numéro ou adresse et un contenu. Ici, l'adresse est toujours un nombre pur tandis que le contenu est un octet particulier (caractère numérique, alphabétique ou spécial).

La façon courante de désigner une zone de mémoire est de spécifier l'adresse de l'octet de gauche et sa longueur. Dans l'exemple suivant

E	L	E	M	E	N	T	S		D	'	I	N	F	O	R	M	A	T	I	Q	U	E

adresse: 2000 2001 2002 ...

l'information est située en mémoire à partir de l'adresse 2000 et occupe 23 octets consécutifs.

Les deux grands principes que nous énoncerons maintenant doivent être retenus à cause de leur implication dans la programmation d'un ordinateur.

Principe 1: La mémoire retient une information tant et aussi longtemps qu'on ne la recouvre pas par une autre.

Suite à l'exemple de la page précédente, introduisons maintenant les 8 octets [H][I][S][T][O][I][R][E] à partir de l'adresse 2011. Le résultat sera:

[E][L][E][M][E][N][T][S][ ][D]['][H][I][S][T][O][I][R][E][I][Q][U][E]

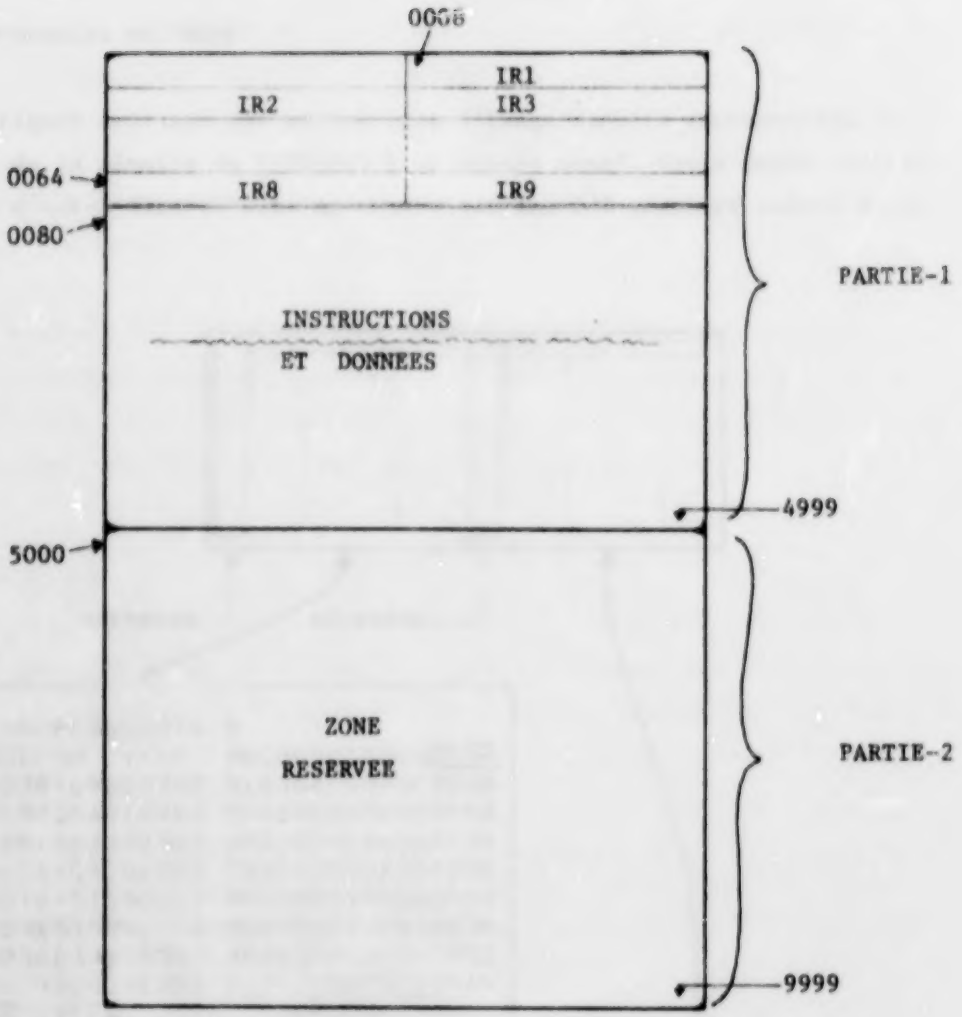
Parce que le "IQUE" n'a pas été recouvert, il demeure intact en mémoire. Pour le faire disparaître, il aurait fallu introduire les 12 octets [H][I][S][T][O][I][R][E][ ][ ][ ][ ] à partir de l'adresse 2011.

Principe 2: L'extraction d'une information de la mémoire n'est pas destructrive.

Ce principe a comme conséquence que si, par exemple, on déplace 10 octets à partir de l'adresse 2000 pour les emmagasiner à partir de l'adresse 2050, la même information se retrouve aux deux endroits après le déplacement.

2.3. La mémoire du FICTIF-2.

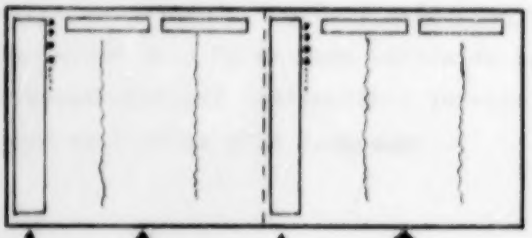
La mémoire du FICTIF-2 contient 10,000 octets adressables individuellement. L'illustration de la page suivante donne une idée de son organisation. La deuxième partie de cette mémoire est réservée pour des motifs qui seront donnés plus tard. Quant à la première partie, elle constitue, pour le programmeur, la seule mémoire utilisable et va de l'adresse 0000 jusqu'à l'adresse 4999. Cette première partie se divise en une zone d'index et en une zone réservée aux instructions et aux données. Neuf registres d'index, dont l'utilité apparaîtra dans les chapitres suivants, sont situés des adresses 0008 à 0079 inclusivement. Quant à la zone des instructions et des données, elle va de 0080 à 4999 inclusivement.



L'ensemble des instructions se situe à partir de l'adresse 0080 et occupe un espace variable selon la longueur du programme. Les données occupent alors l'espace restant. La frontière entre les instructions et les données est donc tout à fait variable d'un problème à l'autre.

2.4. Image-mémoire ou "DUMP".

La figure suivante est un spécimen d'image-mémoire représentant le contenu de la mémoire du FICTIF-2 à un moment donné. Cette image est partielle, c'est-à-dire qu'elle ne couvre que les 832 premiers octets de la mémoire.



adresses adresses

```
0000* OOKLMNOPOOOOOO76 R
0064*   ABCDEFGH   .<(+  900000018000[0464]
0128* 5400078408000792 5400080008000008
0192* 0400080854010468 0401046454000808
0256* 5000081608000008 2700016056000792
0320* 2700014454000705 7900070454000800
0384* 0400072491000002 8000070456000008
0448* 2700036035000000 0000000201010230
0512* 2222230233333333 3520356542325452
0576* 0000023200022222 2120272223023565
0640*    [NON TRIES]          TRIES
0704*                         3333
0768*                      0000007200000000
```

```
0032*
0096* [9]1000002800000624 5400046480000544
0160* 5601046404010468 2300023254010464
0224* 0401046856000792 0800000822000280
0288* 0800080022000328 5100081608000792
0352* 0800000854010464 0400073854010544
0416* 0800082422000456 5000081608000008
0480* 0232050212342023 2032212022222222
0544* 1234023020320101 0502423254523520
0608* 2222202333333333
0672*
0736*    5452
0800* 000000002032       0000000400000076
```

Les octets sont regroupés 16 par 16 sur ce document pour en faciliter la lecture. Chaque ligne complète représente donc 64 octets consécutifs de la mémoire. A chaque 32 octets, une adresse est donnée pour aider le repérage.

Ex: de 0642 à 0650, on trouve 'NON TRIES';
de 0092 à 0097, on trouve '046491'.

Ce document est fourni sur demande, après l'exécution d'un programme, par un programme spécial situé dans la seconde partie de la mémoire.

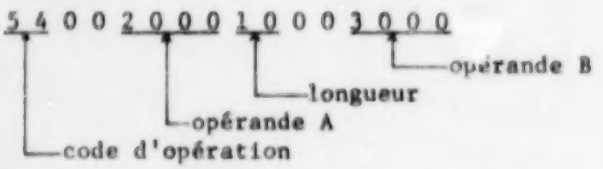
L'utilité d'une image-mémoire est indiscutable lorsque vient le temps de rechercher les erreurs de logique d'un programme.

2.5. Instructions.

La zone 0080-4999 de la mémoire contient les instructions et les données d'un programme. Il est aisé de saisir qu'une donnée est un nombre, un mot, un groupe de mots ou un groupe particulier de symboles. Mais à quoi ressemble une instruction?

Déf: **Une instruction, dans le cas du FICTIF-2, est un groupe de 8 ou de 16 chiffres dont les deux premiers représentent le code d'opération, c'est-à-dire le genre d'opération à exécuter, et dont les autres chiffres constituent des informations essentielles à son exécution.**

Ex:



La signification de cette instruction est la suivante:
Déplacer (code 54) 10 octets de la zone commencant à l'adresse 2000 vers la zone commencant à l'adresse 3000.
Avant l'opération:

Après l'opération:

Nous avons déjà mentionné que, pour qu'un programme s'exécute, les instructions de ce programme devaient déjà se trouver dans la mémoire. La raison en est simple: s'il fallait que chaque instruction arrive une à une de l'extérieur, le rythme d'exécution serait limité par la cadence d'entrée des instructions. Parce que la mémoire est un dispositif capable de fournir une information des milliers de fois plus rapidement que, par exemple, un lecteur de cartes, la vitesse d'exécution peut devenir très élevée. Dans les ordinateurs les plus puissants, le rythme d'exécution est de quelques millions d'instructions à la seconde. Dans une machine plus modeste, il est de quelques centaines à quelques milliers d'instructions à la seconde.

Déf: Un programme constitué d'instructions telles que définies plus haut est appelé programme en langage machine parce que les instructions écrites sous cette forme sont directement reconnaissables par l'unité de contrôle.

Chapitre 3.

Eléments de programmation en langage machine.

Nous présentons ici les principaux codes d'opération en ayant recours à deux exemples de programme.

3.1. Codes d'opération (en partie).

Pour le premier programme, à la section 3.2, nous utiliserons les codes d'opération donnés dans le tableau qui suit. Les opérandes A et B sont les zones de mémoire sur lesquelles portent une opération. Leurs adresses sont données dans l'instruction de même qu'une longueur commune à ces deux opérandes. La longueur des opérandes est limitée: dans le cas d'une opération arithmétique, le maximum est de 16 octets; dans celui d'un déplacement d'information d'une zone à l'autre de la mémoire, la longueur maximale des opérandes est de 80 octets.

Code d'opération	Signification	Longueur de l'instruction	Longueur des opérandes
50	A + B → B	16	01-16
51	B - A → B	16	01-16
52	A × B → B	16	01-16
53	B ÷ A → B	16	01-16
54	A → B	16	01-80
90	Lire et emmagasiner → B	16	80
91	Ecrire le contenu de B	16	80
35	Arrêt	8	nil

Note: la dernière colonne de ce tableau indique les limites inférieures et supérieures: toutes les valeurs comprises entre ces deux nombres inclusivement (s'il y a lieu) sont permises.

3.2. Un programme simple.

Les données que traitent un programme ont deux sources bien précises:

a) elles peuvent être déjà en mémoire au même titre que les instructions avant le début de l'exécution du programme;

b) elles peuvent être introduites dans la mémoire après le début de l'exécution du programme avec une instruction de lecture. Cette lecture s'effectue alors à partir d'un périphérique, soit le lecteur de cartes, soit un dérouleur de bandes magnétiques.

Problème à résoudre:

Il faut lire une carte de donnée comportant 80 colonnes dont les 3 premières contiennent un nombre pair et dont les autres sont libres, diviser ce nombre par 2 et imprimer la valeur originale et le résultat sur une même ligne de l'imprimante (80 colonnes également).

adresses	instructions	
0080	90 00 0001 80 00 1000	(1)
0096	54 00 1000 03 00 1020	(2)
0112	53 00 0900 03 00 1020	(3)
0128	91 00 0002 80 00 1000	(4)
0144	35 00 0000	(5)
0900	002	(6)

Note: les vides laissés dans les instructions ne sont là que pour en faciliter la lecture.

Explication des instructions:

(1) Lire (90) dans le lecteur de cartes (désigné par 0001), une carte de 80 colonnes et emmagasiner son contenu en mémoire centrale à partir de l'adresse 1000. Le choix des adresses 1000 à 1079 pour emmagasiner le contenu de la carte est ab-

solument arbitraire. Cette zone doit être cependant assez éloignée pour que les instructions puissent toutes la précéder.

(2) Déplacer (54) les 3 octets situés à partir de l'adresse 1000 vers la zone commençant à l'adresse 1020. Notez qu'après l'exécution de cette instruction, la même valeur se retrouve aux deux endroits.

(3) Diviser (53) le nombre de 3 chiffres situé à partir de l'adresse 1020 par celui situé à partir de 0900 (ce nombre, comme le montre la ligne (6), est 002). Le résultat de la division sera situé à partir de l'adresse 1020.

(4) Ecrire (91) sur l'imprimante (désignée par 0002) les 80 colonnes situées à partir de l'adresse 1000.

(5) Arrêt (35).

(6) 002. Il s'agit ici d'une information en mémoire avant l'exécution. D'ailleurs, toute la partie encadrée représente ce qui doit être en mémoire centrale avant le début de l'exécution du programme.

3.3. Codes d'opération (en partie).

Si tous les programmes devaient se limiter à une suite absolument linéaire d'instructions, l'ordinateur n'aurait pas la souplesse qu'on lui connaît. Cette souplesse vient de la possibilité d'effectuer des comparaisons entre des valeurs en mémoire et, selon le résultat du test, de continuer l'exécution du programme à telle ou telle instruction. Cette opération se fera toujours en deux temps:

a) dans une instruction de comparaison, on détermine, entre deux valeurs, si la première est plus petite, égale ou plus grande que la deuxième. Ce résultat est retenu par la machine.

b) par une instruction de saut conditionnel, on demande de sauter à une instruction donnée si la condition est réalisée. Si elle ne l'est pas, le programme se continue selon la séquence normale.

Par exemple, en b) on demande de sauter conditionnellement à une certaine instruction si, lors de la dernière comparaison, le premier nombre était plus petit que le second. Si, de fait, la comparaison en a) montrait une égalité, il est clair que le saut conditionnel ne se fera pas car la condition demandée n'est pas réalisée. Le programme, dans ce cas, se poursuivra séquentiellement.

Code d'opération	Signification	Longueur de l'instruction	Longueur des opérandes
56	Comparer A et B	16	01-16
21	Saut si A était < B	8	nil
22	" " A était = B	8	"
23	" " A était ≤ B	8	"
24	" " A était > B	8	"
25	" " A était ≠ B	8	"
26	" " A était ≥ B	8	"
27	saut inconditionnel	8	"

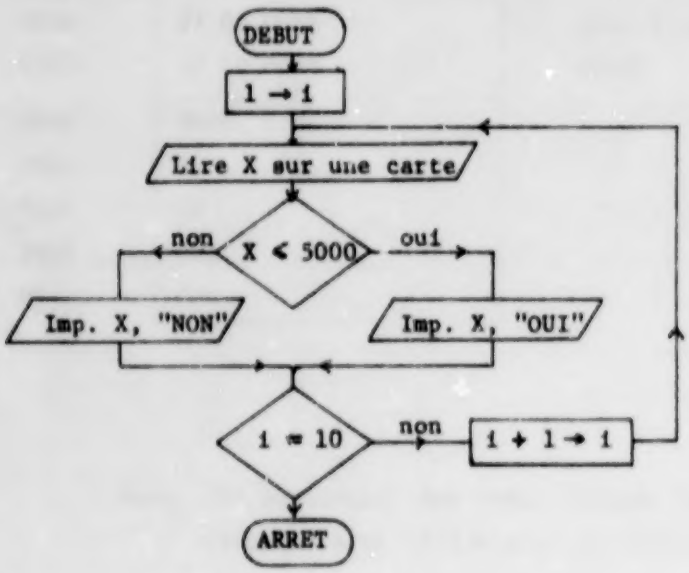
Le saut inconditionnel (27) n'est lié à aucune condition et se fait nécessairement. L'opérande A d'une instruction de saut est l'adresse de l'instruction où le saut doit se faire.

3.4. Un autre programme simple.

Problème à résoudre:

Lire 10 cartes les unes après les autres. Chaque carte contient un nombre de 4 chiffres dans les colonnes 1 à 4 et est libre quant au reste; il faut déterminer si ce nombre est plus petit ou égal à 5000. S'il l'est, on imprime "OUI" à côté du nombre lu. Sinon, on imprime "NON".

Voici un organigramme satisfaisant:



Choix (arbitraire) des adresses à utiliser:

adresse ←→	contenu
1000-1079	carte lue (X : 1000-1003)
1100-1101	i
0900-0903	5000
0904-0905	01
0906-0907	10
0908-0910	OUI
0911-0913	NON

adresse		
0080	54 00 0904 02 00 1100	1 → i
0096	90 00 0001 80 00 1000	lire une carte
0112	56 00 1000 04 00 0900	comparer X et 5000
0128	23 00 0160	si ≤ , aller à 0160
0136	54 00 0911 03 00 1010	mettre "NON" près de X
0152	27 00 0176	saut pour imprimer
0160	54 00 0908 03 00 1010	mettre "OUI" près de X
0176	91 00 0002 80 00 1000	imprimer
0192	56 00 1100 02 00 0906	comparer i et 10
0208	22 00 0240	si = , saut à arrêt
0216	50 00 0904 02 00 1100	add. 1 à i
0232	27 00 0096	saut à la lecture
0240	35 00 0000	arrêt
0900	5000	
0904	01	
0906	10	
0908	OUI	
0911	NON	

Note: ici également, les vides laissés dans les instructions ne sont là que pour en faciliter la lecture.

Chapitre 4.

Les registres d'index et l'incrémentation automatique.

Dans l'ordinateur FICTIF-2, neuf (9) registres d'index sont disponibles. Ils occupent la zone de mémoire allant des adresses 0008 à 0079. Chaque registre d'index comporte 8 octets dont l'adresse du premier, divisé par 8, donne le numéro du registre. Ainsi le registre d'index situé de 0048 à 0055 porte le numéro 6.

4.1. Rôle des registres d'index.

Le premier exemple d'instruction que nous ayons présenté dans ce texte, à la section 2.5, était le suivant:

```
54 00 2000 10 00 3000
   └──────┘   └──────┘──opérande B
      └──opérande A
```

Aucun détail n'avait été donné sur la raison d'être des quatrième et douzième chiffres. De fait, ils servent à désigner un numéro d'index entre 1 et 9. Un 0 indique évidemment qu'aucun registre d'index n'intervient.

Lorsqu'un registre d'index est désigné dans un opérande, le contenu des 4 derniers octets de ce registre s'ajoute à l'adresse de base donnée pour former l'adresse effective.

Supposons l'instruction suivante

```
54 02 2000 10 00 3000
    └──registre d'index 2
```

avec le registre 2 (0016-0023) = 00000020,

alors l'adresse effective du premier opérande sera:

adresse de base + contenu du registre d'index 2 = 2000 + 0020
= 2020

Déf: Un registre d'index constitue un moyen de modifier l'adresse d'un opérande dans une instruction, juste avant son exécution, sans modifier l'apparence extérieure de cette instruction.

Le calcul des adresses effectives, pour une instruction, se fait entre le moment où l'unité de contrôle extrait cette instruction de la mémoire en vue de son exécution et celui où elle est exécutée.

4.2. Exemple d'utilisation de registre d'index.

Problème à résoudre:

Une suite de 20 éléments de quatre octets chacun se trouve en mémoire à partir de l'adresse 3000. Par ailleurs nous avons un élément X de 4 octets et nous désirons savoir s'il appartient ou non à cette suite. Dans une adresse particulière Y, nous mettrons 1 ou 0 selon qu'il aura été trouvé ou pas.

Solution:

Il s'agit évidemment de comparer successivement X aux éléments de la suite situés à 3000, 3004, 3008, ... , 3076.

Nous suggérons la méthode suivante:

Dans une boucle de programme, donnons au registre d'index 1, IR1, les valeurs successives

0000, 0004, 0008, ... , 0076 qui viendront s'ajouter à l'adresse de base 3000.

Choix des adresses à utiliser:

adresse ⟷	contenu
1000-1003	0000
1004	Y
1005-1008	X
3000-3079	suite des 20 nombres
1009	0
1010	1
1011-1014	0004
1015-1018	0076

adresse		
0080	54 00 1000 04 00 0012	0 → IR1
0096	56 00 1005 04 01 3000	comp. X et la suite indexée
0112	25 00 0144	si ≠ , aller à 0144
0120	54 00 1010 01 00 1004	1 → Y
0136	27 00 0216	saut à arrêt
0144	56 00 0012 04 00 1015	comp. IR1 et 0076
0160	25 00 0192	si ≠ , aller à 0192
0168	54 00 1009 01 00 1004	0 → Y
0184	27 00 0216	saut à arrêt
0192	50 00 1011 04 00 0012	add. 0004 à IR1
0208	27 00 0096	saut à 0096
0216	35 00 0000	arrêt
1000	0000	
1009	0100040076	

Explication:

A la première exécution de l'instruction

56 00 1005 04 01 3000
(01 3000 : opérande B)

l'adresse effective de l'opérande B est

3000 + contenu de IR1 = 3000 + 0000 = 3000.

Avant la deuxième exécution de cette instruction, le registre d'index aura été augmenté de 0004 (à l'instruction située à 0192). Donc, l'adresse effective de l'opérande B deviendra

3000 + contenu de IR1 = 3000 + 0004 = 3004.

En continuant ainsi, il est clair que X aura été comparé à chaque élément de la suite, du moins jusqu'à temps d'en trouver un qui lui soit égal.

4.3. Règles d'utilisation des registres d'index.

Répétons que, même si les registres d'index occupent 8 octets, seulement les 4 derniers sont utilisés dans le calcul des adresses. Nous dirons qu'à toute fin pratique,

le registre 1 (IR1) est situé de 0012 à 0015,
" 2 (IR2) " 0020 à 0023,
.
.
.
" 9 (IR9) " 0076 à 0079.

Il en découle que, lorsque nous voulons effectuer des opérations arithmétiques, de déplacement, de comparaison avec un registre d'index, il suffit d'utiliser ces quatre octets de droite.

Les opérandes A et B d'une instruction peuvent faire usage en même temps du même registre d'index ou de deux registres différents.

IMPORTANT: un registre d'index utilisé dans une instruction doit toujours avoir une valeur numérique, sans quoi l'exécution du programme bloquera sur cette instruction.

4.4. Incrémentation automatique.

Il s'agit ici d'une variante dans l'utilisation des registres d'index où il n'est plus nécessaire de programmer l'addition d'une valeur au registre d'index pour lui faire désigner les éléments successifs d'une suite.

> Dans l'incrémentation automatique, lors du calcul de l'adresse effective d'un opérande, la somme de l'adresse de base et du registre d'index est immédiatement reportée dans le registre en plus de devenir adresse effective.

Dans une instruction, l'utilisation de l'incrémentation automatique peut toucher l'opérande A et/ou B. Elle est signalée par un 1 en position 3 et/ou 11 de cette instruction.

Ex: 54 12 0004 04 00 2000

registre 2 (= 00002996, par hypothèse).

incrémentation automatique signalée.

L'adresse effective de l'opérande A sera

0004 + contenu de IR2 = 0004 + 2996

= 3000

mais en plus ce 3000 sera immédiatement replacé dans le registre 2.

Une deuxième exécution de cette instruction donnerait comme adresse effective pour l'opérande A

0004 + contenu de IR2 = 0004 + 3000

= 3004

avec 3004 emmagasiné en plus dans ce registre.

Le but de l'incrémentation automatique est d'éliminer des instructions et d'augmenter la rapidité d'exécution du programme.

4.5. Exemple d'utilisation de l'incrémentation automatique.

Problème à résoudre:

Nous reprenons le problème posé à la section 4.2.

Solution:

Parce que les éléments ont 4 octets de long, il faut, au point de départ, mettre dans le registre d'index choisi l'adresse du début de la suite moins 4, c'est-à-dire 3000 - 4 = 2996.

Choix des adresses à utiliser:

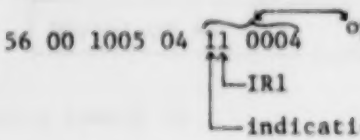
adresse ⟷	contenu
1000-1003	2996
1004	Y
1005-1008	X
3000-3079	suite de 20 nombres
1009	0
1010	1
1011-1014	3076

adresse		
0080	54 00 1000 04 00 0012	2996 → IR1
0096	56 00 1005 04 11 0004	comp. X et un élément
0112	25 00 0144	si ≠ , aller à 0144
0120	54 00 1010 01 00 1004	1 → Y
0136	27 00 0184	saut à arrêt
0144	56 00 0012 04 00 1011	comp. IR1 et 3076
0160	25 00 0096	si ≠ , aller à 0096
0168	54 00 1009 01 00 1004	0 → Y
0184	35 00 0000	arrêt
1000	2996	
1009	013076	

Le programme comporte 3 instructions de moins que le précédent.

Explication:

A la première exécution de l'instruction



l'adresse effective de l'opérande B est

0004 + contenu de IR1 = 0004 + 2996 = 3000.

Comme 3000 a été automatiquement remis dans IR1, l'adresse effective de l'opérande B sera, à la deuxième exécution de l'instruction,

0004 + contenu de IR1 = 0004 + 3000 = 3004.

Donc, en continuant ainsi, chaque élément de la suite pourra éventuellement être comparé à X.

Chapitre 5.

Les périphériques de l'ordinateur FICTIF-2.

Un lecteur de cartes, une imprimante et quatre dérouleurs de bandes magnétiques constituent l'ensemble des périphériques de l'ordinateur FICTIF-2.

PERIPHERIQUES

Nom	No de l'unité	
Lecteur de cartes	1	
Imprimante	2	
Dérouleur #1	3	
Dérouleur #2	4	
Dérouleur #3	5	} Réservés au système.
Dérouleur #4	6	

5.1. Le lecteur de cartes (unité 1).

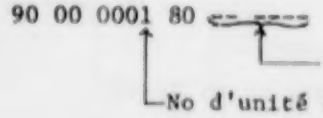
Il lit les cartes régulières de 80 colonnes et reconnaît les caractères suivants:

```
0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L
M N O P Q R S T U V W X Y Z . , - $ ç * / +
: = ' ( ) < >
```

Chaque caractère est conservé en mémoire à l'aide d'une configuration de huit bits, selon le code interne défini antérieurement. Si un caractère inconnu apparaît sur la carte, il sera remplacé par le caractère <.

En utilisant la perforatrice IBM 29, la touche ç doit être utilisée pour produire le symbole ç en mémoire.

Le lecteur de cartes est chargé de l'exécution de l'instruction



La réaction normale du lecteur de cartes à cette instruction est de lire une carte et d'en communiquer le contenu à la mémoire qui l'emmagasine à l'adresse indiquée par l'opérande B. Il y a des cas, lors d'un bris de carte par exemple, où le lecteur détecte une anomalie. Un message d'état est toujours retourné en mémoire centrale à l'adresse 0000.

Message d'état	Signification
0	Opération complétée correctement.
1	Opération complétée mais au moins un caractère inconnu a été remplacé par < .
2	Défaillance physique. Opération non complétée.
3	Lecture impossible: lecteur de cartes vide.

Dans un chapitre ultérieur, nous verrons l'utilité d'un tel message.

5.2. L'imprimante (unité 2).

L'imprimante est capable d'imprimer tous les caractères acceptés par le lecteur de cartes. Si, par suite de manipulations internes, un octet non imprimable est généré, l'imprimante le remplacera par un blanc au moment de l'impression.

Il est à remarquer que cette imprimante est limitée à des lignes de 80 colonnes.

L'imprimante est chargée de l'exécution des instructions suivantes:

a) 91 00 0002 80 — opérande B; No de périphérique.

b) 31 00 0002

Le message d'état de l'imprimante est retourné en mémoire centrale à l'adresse 0001.

Message d'état	Signification
0	Opération complétée correctement.
1	Manque de papier.
2	Défaillance physique. Opération non complétée.

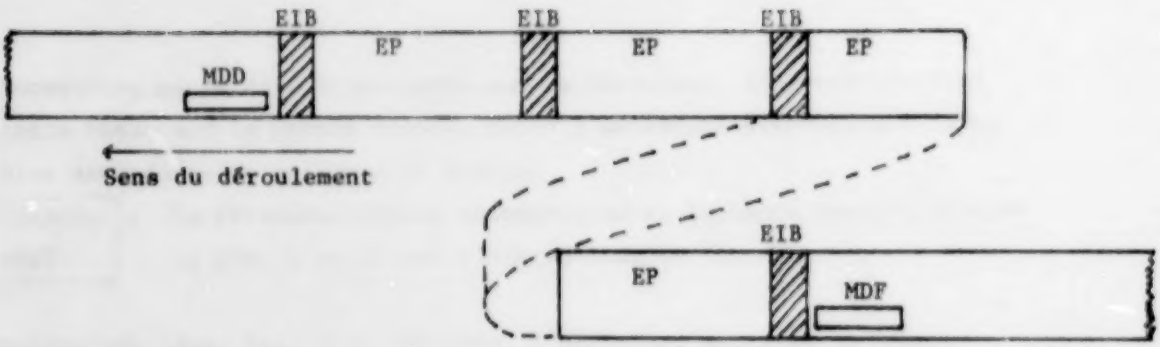
5.3. Les dérouleurs de bandes magnétiques (Unités 3, 4, 5, 6).
5, 6 — réservés.

*** Bandes magnétiques.

Une bande magnétique est un support séquentiel d'information. Elle se présente normalement en longueur standard. Il existe cependant des longueurs différentes, autorisant des capacités de stockage variables.

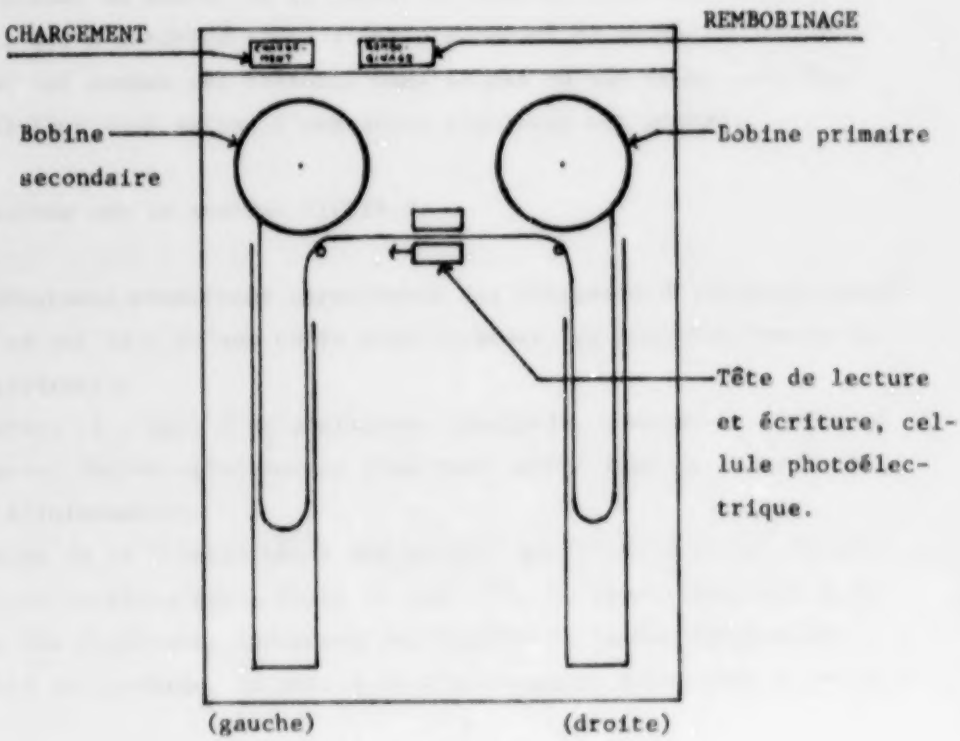
L'information est regroupée en enregistrements physiques. On appelle enregistrement physique (EP) l'ensemble des informations transmises dans une même commande de lecture ou d'écriture. Normalement, le dérouleur s'arrête entre chaque opération d'entrée ou de sortie de sorte qu'on doit prévoir entre deux enregistrements physiques un espace de décélération et d'accélération appelé "espace inter-bloc" ou EIB.

L'illustration suivante montre comment les enregistrements se succèdent sur une bande:



On note la présence au début et à la fin du ruban de deux petites surfaces marquant le début (Marque de début: MDD) et la fin (Marque de fin: MDF) du ruban. Ces deux surfaces collées sur le ruban réfléchissent la lumière et le dérouleur peut en détecter la présence à l'aide d'une cellule photoélectrique.

*** Dérouleur de bandes.



Lorsqu'une bande doit être montée sur un dérouleur, l'opérateur place cette bande sur la partie droite, relie l'extrémité disponible à la bobine secondaire et actionne le bouton:

CHARGE-MENT

. Le dérouleur avance automatiquement la bande jusqu'à trouver le MDD. L'unité est alors en état de fonctionner.

Lorsqu'une bande doit être démontée, l'opérateur actionne le bouton:

REMBO-BINAGE

. La bande s'enroule alors à haute vitesse sur la bobine primaire jusqu'à ce que le dérouleur détecte le MDD. Il y a alors arrêt et l'opérateur, en actionnant une seconde fois le même bouton, provoque la fin de l'enroulement mais à vitesse réduite cette fois.

Le dérouleur est muni d'un dispositif de protection intéressant. La bobine originale (primaire) possède un anneau de plastique qui peut être présent ou absent au moment où la bande est montée. Cela est détecté par le dérouleur qui n'autorise alors l'enregistrement de nouvelles informations que si cet anneau est présent. Dans le cas où une bande doit être protégée, il faut donc aviser l'opérateur d'enlever cet anneau.

*** Restrictions sur le système FICTIF-2.

Les ordinateurs permettent normalement des longueurs d'enregistrements variables, ce qui fait qu'une bande peut posséder des enregistrements de longueurs différentes.

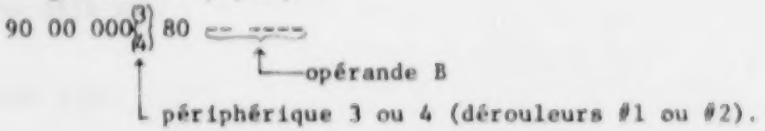
Ici, puisqu'il s'agit d'un ordinateur simplifié, nous avons déterminé que la longueur des enregistrements physiques serait fixe et égale à 80 caractères d'information.

Les bandes de la "bibliothèque des bandes" du FICTIF-2 seront identifiées par deux chiffres entre 01 et 99 (ex: 45). Un répertoire, mis à la disposition des étudiants, indiquera les numéros de bandes existantes, leur capacité de stockage, la nature de l'information qu'on peut y retrouver.

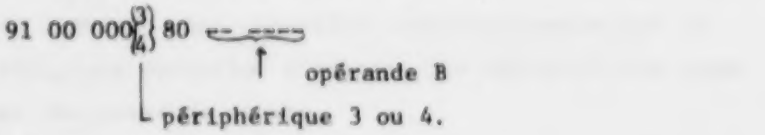
*** Opérations permises par programmation.

Si on exclut l'utilisation des unités 5 et 6, les dérouleurs #1 et #2 (unités 3 et 4) sont chargés de l'exécution des instructions suivantes:

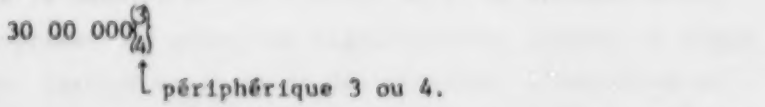
a) Lire un enregistrement physique.



b) Ecrire un enregistrement physique.



c) Rembobiner le ruban.



Dans ce cas, il est possible de reprendre le traitement de la bande sans interrompre le programme.

Le <u>message d'état</u> suivant l'exécution d'une de ces instructions est placé en mémoire aux adresses 0002 et 0003. L'adresse 0002 contient le numéro du périphérique concerné (ici 3 ou 4) et l'adresse 0003 contient le message lui-même:

Message d'état	Signification
0	Opération complétée correctement.
1	Aucune bande sur cette unité.
2	Marque MDF rencontrée.
3	Rembobinage refusé p.c.q. aucu- bande sur cette unité.
4	Refus d'écrire. Bande protégée, anneau absent.

Chapitre 6.

Le langage machine du FICTIF-2. Compléments.

La plupart des codes d'opération ont été abordés aux chapitres 3 et 5. Nous expliquerons ici les codes qui restent et apporterons des précisions sur certains codes déjà vus.

6.1. L'instruction d'édition (55).

Dans la plupart des cas, les champs numériques qui ont servis dans les opérations arithmétiques doivent, pour apparaître avantageusement sur un rapport d'impression, subir une opération d'édition par laquelle, par exemple, on pourra supprimer les zéros de gauche.

L'édition est une opération qui permet de déplacer un champ numérique d'un lieu à un autre de la mémoire en lui faisant subir un certain nombre de transformations: supprimer les zéros non significatifs, ajouter un signe de dollar sur la gauche, insérer un point et des virgules. L'exécution de cette instruction se fait sous le contrôle d'un masque d'édition situé dans la zone réceptrice et ayant même longueur que la zone émettrice à éditer. Le masque d'édition se détruit au fur et à mesure que l'instruction s'exécute. Le déplacement se fait de droite à gauche.

ex 1: 55 00 1000 06 00 1500

Avant l'exécution:

1000	1500
0 0 0 1 5 0	Z Z Z Z Z 9

Après l'exécution:

1000	1500
0 0 0 1 5 0	1 5 0

Notes:
1. Un 9 dans le masque indique que le chiffre correspondant de la zone émettrice sera déplacé tel quel.
2. Un Z dans le masque indique que le chiffre correspondant de la zone émettrice, s'il est un zéro non significatif, sera remplacé par un ∅ . Sinon, le Z est considéré comme un 9.

3. Les autres caractères permis dans un masque d'édition sont le point (.), la virgule (,), et le signe de dollar ($).
4. Si la zone émettrice n'est pas numérique ou si le masque contient autre chose qu'une combinaison de (9), (Z), (,), (.) ou ($), l'exécution du programme sera abandonnée et un message d'erreur sera imprimé.

ex 2: 55 00 1000 08 00 1500

Avant l'exécution:

1000	1500
0 0 0 0 2 1 5 0	Z Z Z Z 9 . 9 9

Après l'exécution:

1000	1500
0 0 0 0 2 1 5 0	2 1 . 5 0

ex 3: 55 00 1000 07 00 1500

Avant l'exécution:

1000	1500
0 0 0 2 1 5 0	$ $ $ 9 . 9 9

Après l'exécution:

1000	1500
0 0 0 2 1 5 0	$ 2 1 . 5 0

ex 4: 55 00 1000 10 00 1500

Avant l'exécution:

1000	1500
0 0 0 0 4 2 3 1 5 0	$ $ $, $ $ 9 . 9 9

Après l'exécution:

1000	1500
0 0 0 0 4 2 3 1 5 0	$ 4 , 2 3 1 . 5 0

Il est à remarquer que dans les exemple 3 et 4, les signes de $ coincidant avec des chiffres significatifs sont considérés comme des 9. De plus, un seul $ peut être présent dans le résultat final.

6.2. Saut avec adresse de retour (58).

D'une importance capitale lorsqu'on veut utiliser une sous-routine (séquence d'instructions dont on demande l'exécution à partir de plusieurs points du programme), cette instruction permet d'effectuer un saut tout en conservant dans un registre d'index désigné, l'adresse de la prochaine instruction, appelée adresse de retour. La reprise du programme principal à l'endroit où il avait été laissé est, de ce fait, grandement facilitée.

Voici un exemple d'utilisation:

Une sous-routine est écrite pour faire sauter un nombre particulier X (une valeur entre 1 et 9) de lignes blanches. Elle sera située à partir de l'adresse 4000. Cette sous-routine est appelée de divers points du programme principal.

Choix des adresses:

1000	2
1001	5
4800	X (nombre de lignes à sauter)
4811-4890	zone de 80 blancs
4900	1
4901	1

Programme principal:

0080	...	
	...	
0200	54 00 1000 01 00 4800	2 → X (pour faire sauter 2 lig.)
0216	58 00 0002 00 00 4000	Sauter à l'adresse 4000 après avoir mis l'adresse de la prochaine inst. (0232) dans IR2.
0232	...	
	...	
0400	54 00 1001 01 00 4800	5 → X (pour faire sauter 5 lig.)
0416	58 00 0002 00 00 4000	Sauter à 4000 après avoir mis 0432 dans IR2.
0432	...	
	...	

Sous-routine:

4000	54 00 4900 01 00 4901	1 → 1
4016	90 00 0002 80 00 4811	imprimer une ligne blanche.
4032	56 00 4901 01 00 4800	comparer 1 et X.
4048	22 00 4080	si = , saut à 4080.
4056	50 00 4900 01 00 4901	add. 1 à 1.
4072	27 00 4016	saut à 4016
4080	27 02 0000	saut à l'adresse contenue dans IR2.

Explication:

A l'adresse 0216, la sous-routine est appelée, c'est-à-dire qu'un saut est fait à l'adresse 4000. La sous-routine s'exécute et imprime deux fois une ligne blanche après quoi se présente l'instruction

27 02 0000

↑—adresse effective =

0000 + contenu du registre 2 = 0000 + 0232

= 0232.

ce qui correspond à un saut à l'adresse suivant l'instruction qui a appelé cette sous-routine.

Il est clair que cette sous-routine peut être appelée de n'importe quel endroit du programme principal et on est sûr que celui-ci se continuera séquentiellement après l'exécution de cette sous-routine. Autrement dit, le retour se fait à l'instruction suivant l'appel de la sous-routine.

6.3. Précisions sur l'addition et la soustraction.

L'unité arithmétique et logique effectue une addition, octet par octet, de droite à gauche, sur les opérandes A et B en emmagasinant le résultat dans l'opérande B. Dans cette opération, les 4 bits de gauche de chaque octet sont ignorés bien que ceux-ci soient toujours ajustés à 0011 dans l'octet résultant.

Dans les exemples qui suivent, chaque octet occupe un petit rectangle dont le contenu constitue sa représentation interne selon le code défini dans un chapitre précédent. Cependant, au lieu d'écrire 4 bits, on préférera utiliser le caractère hexadécimal correspondant.

ex 1: 4 + 2 = 34
+ 32
36 = 6

ex 2: B + 2 = 42
+ 32
34 = 4

ex 3: 14 + 18 = 31 34
+ 31 38
33 32 = 32

ex 4: 4 + 8 = 34
+ 38
32 = 2 avec report.

ex 5: 44 + 88 = 34 34
+ 38 38
33 32 = 32 avec report.

L'exemple 2 illustre le fait que les 4 premiers bits d'un octet sont ignorés. En effet, la lettre B (42) revient au même que 2 (32). Bien que les circuits autorisent un opérande contenant des caractères autres que des chiffres, c'est une erreur de programmation que de procéder à une addition avec des opérandes non numériques.

Si l'addition se termine avec un report comme dans les exemples 4 et 5, l'indicateur DEPASSEMENT est mis à 1. Sinon, il est laissé à 0, son état normal.

L'unité arithmétique et logique effectue une soustraction, octet par octet, de droite à gauche. L'opérande A est soustrait de l'opérande B qui retient le résultat. Les 4 bits de gauche de chaque octet résultant sont automatiquement ajustés à 0011. l'opération est conduite comme suit:

en partant de la droite, le complément de 10 d'un octet de l'opérande A est additionné à l'octet équivalent de l'opérande B. Si le résultat est plus petit ou égal à 9, un emprunt est signalé pour le prochain octet sur la gauche. L'opération se poursuit ainsi jusqu'à la fin. Si le calcul se termine avec un emprunt sur la gauche, il faut interpréter le résultat comme étant sous forme complémentaire et l'indicateur DEPASSEMENT est mis à 1: dans le cas contraire, il est laissé à 0.

Voici quelques exemples de soustraction en utilisant les mêmes conventions qu'à la page précédente:

ex 1: 25 - 16 = [32] [35] (B=25, A=16)

- [31] [36]

= [32] [35]

+ [39] [34]

[30] [39] = 09, résultat final.

ex 2: 26 - 34 = [32] [36]

- [33] [34]

= [32] [36]

+ [37] [36]

[39] [32] = 92 avec un emprunt final, ce qui indique un résultat sous forme complémentaire. En effet, le complément de 92 est 08. Donc la réponse sera - 8.

6.4. Saut sur dépassement (20).

Cette instruction est un saut conditionnel qui a la particularité de n'être pas précédé d'une comparaison, mais d'une addition ou d'une soustraction.

Le saut est effectué si l'indicateur DEPASSEMENT a été mis à 1 dans l'instruction d'addition ou de soustraction qui précède immédiatement. Si cet indicateur est toujours à 0, c'est-à-dire qu'il n'a pas été modifié par l'instruction précédente, le programme se poursuit selon la séquence normale.

6.5. Complément d'information sur les opérations arithmétiques et de déplacement.

a) Les opérations de multiplication et de division ne sont pas câblées mais exécutées par programmation. Les opérandes doivent être purement numériques et, dans le cas d'une division, l'opérande A ne doit pas avoir une valeur égale à zéro.

b) Dans une opération de déplacement, les octets sont déplacés d'une zone à l'autre de droite à gauche. Etant donné qu'il n'est pas exclu que ces deux zones puissent se superposer au moins partiellement, nous allons utiliser ce fait , dans l'exemple suivant, pour propager un octet donné dans toute une zone. Pour propager un zéro (0) dans la zone de 80 octets située à 3301-3380, nous allons d'abord placer 0 à l'adresse 3380 et procéder à l'exécution de l'instruction suivante:

54 00 3302 79 00 3301

Dans un premier temps, le contenu de 3380 est mis dans 3379;
dans un deuxième temps, " " 3379 " " 3378;
et ainsi de suite. A la fin, la zone complète sera remplie de zéros.

6.6. Tableau récapitulatif des codes d'opération.

Code d'opération	Signification	Longueur de l'instruction	Longueur des opérandes
50	A + B → B	16	01-16
51	B - A → B	"	"
52	A * B → B	"	"
53	B ÷ A → B	"	"
54	A → B	"	01-80
55	Editer A → B	"	01-16
56	Comp. A et B	"	"
58	Saut ↔ Retour	"	-
20	Saut si Dépas.	8	-
21	Saut si <	"	-
22	" =	"	-
23	" ≤	"	-
24	" >	"	-
25	" ≠	"	-
26	" ≥	"	-
27	Saut incondit.	"	-
30	Rembobiner	"	-
31	Saut de page	"	-
35	Arrêt	"	-
90	Lire	16	80
91	Ecrire	16	80

Chapitre 7.

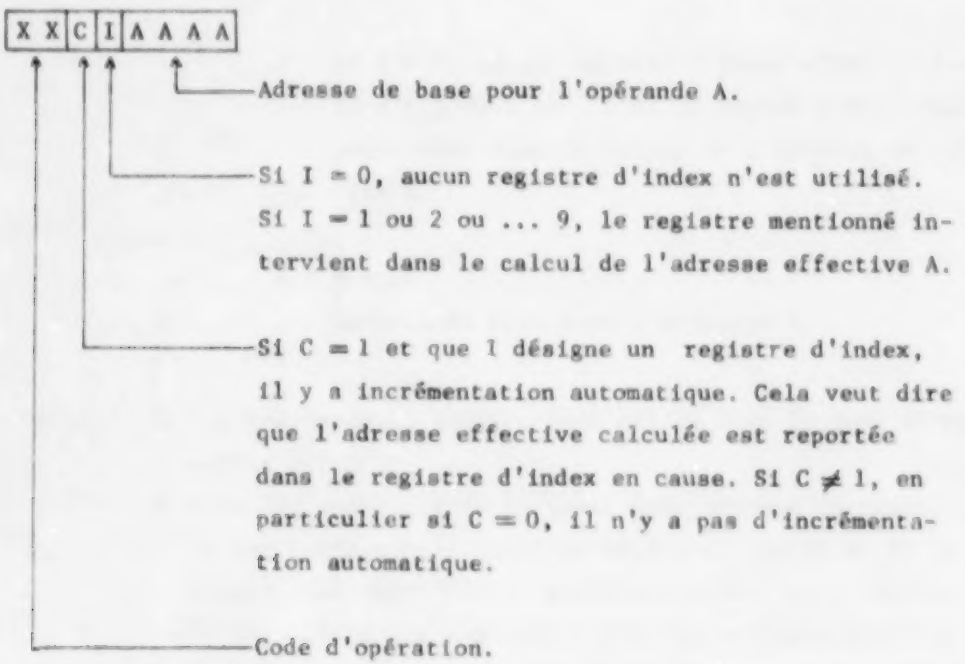
L'unité de contrôle.

Le rôle de l'unité de contrôle du FICTIF-2 a été présenté sommairement au premier chapitre. Rappelons que cette unité dirige l'exécution d'un programme en mémoire. Ici, nous expliciterons davantage les modalités de fonctionnement de cette unité.

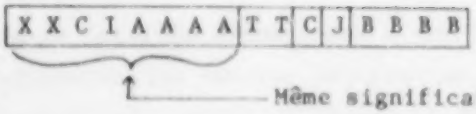
7.1. Formats des instructions.

Toutes les instructions que reconnaît l'unité de contrôle du FICTIF-2 répondent à l'un des deux formats suivants:

a) Format court (8 octets):



b) Format long (16 octets):



Même signification que pour un format court.

T T :

Longueur des opérandes. Cette longueur varie de 01 à 80 selon les instructions. Dans l'instruction 58, elle est inutilisée et mise normalement à 00.

C :

Incrémentation automatique. Les règles sont les mêmes que celles définies à la page précédente pour ce point.

J :

Si J = 0, aucun registre d'index n'est utilisé. Si J = 1 ou 2 ou ... 9, le registre mentionné intervient dans le calcul de l'adresse effective B.

B B B B :

Adresse de base pour l'opérande B.

Notes:

1. Les instructions longues sont celles dont le code d'opération est $\geq$ 50.
2. Dans les instructions 58 (saut avec adresse de retour), 30 (rembobiner), 31 (saut de page), 90 (lire) et 91 (écrire), les troisième et quatrième octets sont ignorés et les octets 5-6-7-8 constituent un opérande immédiat: il ne s'agit pas d'une adresse de la mémoire mais d'une entité directement utilisable par l'unité de contrôle pour identifier un périphérique ou, dans le cas du code 58, un numéro de registre d'index (1 à 9).

7.2. Fonctionnement.

L'unité de contrôle extrait une à une les instructions de la mémoire centrale et active les parties nécessaires à leur exécution. Pour une instruction donnée, ce travail se résume en deux phases:

a) le cycle d'instruction:

L'unité de contrôle consulte son compteur ordinal qui contient l'adresse de la prochaine instruction à exécuter.

ex: compteur ordinal = | 0 | 4 | C | 0 |

Huit octets sont obtenus de la mémoire et le compteur ordinal est augmenté de huit. L'adresse effective de l'opérande A est alors calculée; à ce moment, un registre d'index et l'incrémentation automatique peuvent être pris en considération. Le résultat est que l'adresse effective A est conservée dans un registre spécial de 4 octets:

Op. eff. A (opérande effectif A)

Note: les registres de l'unité de contrôle ne doivent pas être confondus avec les registres d'index. Pour le moment il faut considérer que ces registres sont logés dans l'unité de contrôle même.

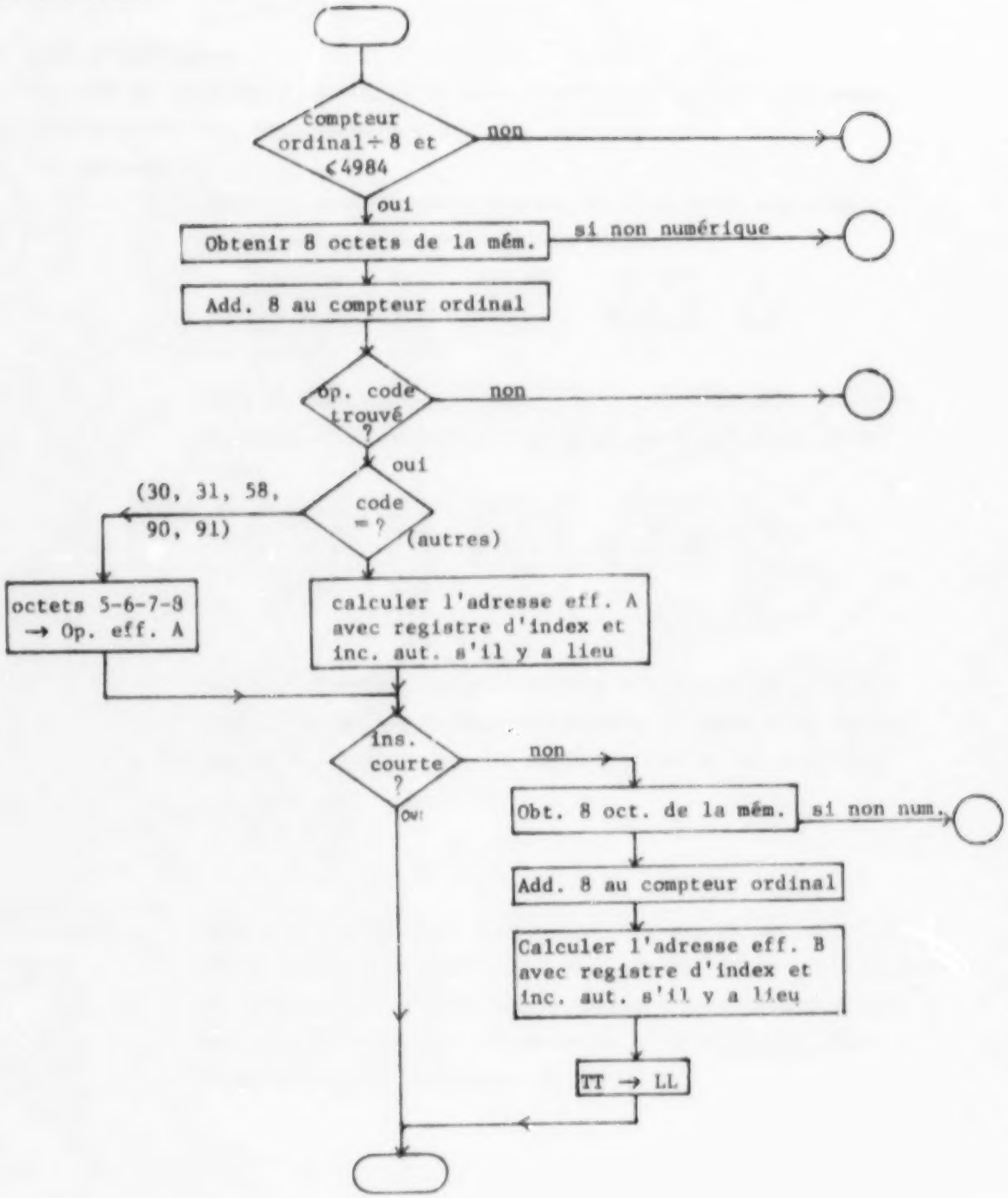
L'unité de contrôle détermine ensuite si le code d'opération de cette instruction indique qu'il s'agit d'une instruction courte ou longue. S'il s'agit d'une instruction longue (code $\geq$ 50), huit autres octets sont obtenus de la mémoire et le compteur ordinal est à nouveau augmenté de huit. L'adresse effective de l'opérande B est alors calculée; ici également, un registre d'index et l'incrémentation automatique peuvent intervenir. Le résultat du calcul est conservé dans un autre registre de 4 octets:

Op. eff. B (opérande effectif B)

Finalement, dans le cas d'une instruction longue, la longueur TT est conservée dans un nouveau registre de 2 octets appelé LL:

LL

Voici un organigramme illustrant la manière dont le cycle d'instruction s'exécute:



Dans la figure précédente, chaque sortie latérale représente un circonstance particulière qui sera traitée dans un chapitre de la partie II.

b) cycle d'exécution:

L'unité de contrôle a identifié le code d'opération et peut maintenant en régir l'exécution. Nous donnerons ici deux exemples:

exemple 1:

Voici le contenu des registres de l'unité de contrôle avant le début du cycle d'instruction:

compteur	op.	op. eff.	op. eff.	LL
0 0 9 6	5 4	2 0 0 0	3 0 0 0	1 0
ordinal	code	A	B	

Soit en mémoire 2 7 0 0 0 4 0 0 à l'adresse 0096. Au cours du cycle d'instruction, les registres s'empliront comme suit:

compteur	op.	op. eff.	op. eff.	LL
0 1 0 4	2 7	0 4 0 0	3 0 0 0	1 0
ordinal	code	A	B	
			intouchés.	

Le cycle d'exécution pour ce type d'instruction consiste, pour l'unité de contrôle, à déplacer le contenu de son registre Op. eff. A dans le compteur ordinal qui deviendra

compteur				
0 4 0 0	...	...	...	...
ordinal				

Comme, après le cycle d'exécution, il y a retour au cycle d'instruction, il est clair que, au cours de ce cycle d'instruction, c'est l'instruction située à l'adresse 0400 qui sera chargée dans les registres de l'unité de contrôle. C'est ainsi que s'effectue un saut.

exemple 2:

Faisons suite à l'exemple précédent et supposons qu'en mémoire, à l'adresse 0400, il y a

5	0	0	1	1	0	0	0	0	5	0	0	2	0	0	0

et que IR1 contient

0	0	0	0	0	0	5	0

.

La première partie du cycle d'instruction amènera le remplissage partiel suivant des registres de l'unité de contrôle:

compteur	op.	op. eff.	op. eff.	LL
0 4 0 8	5 0	1 0 5 0	3 0 0 0	1 0
ordinal	code	A	B	

B : inchangés.

A : 1000 + 0050 = 1050

Comme 50 est le code d'une instruction longue, les 8 octets suivants de la mémoire seront apportés:

compteur	op.	op. eff.	op. eff.	LL
0 4 1 6	5 0	1 0 5 0	2 0 0 0	0 5
ordinal	code	A	B	

Au cycle d'exécution, puisque l'unité de contrôle a reconnu une addition, elle chargera l'unité arithmétique et logique, de pair avec la mémoire qui livrera les données et recevra le résultat, d'effectuer cette opération. Le cycle d'exécution étant terminé, il y aura retour au cycle d'instruction dans lequel sera chargée l'instruction située à l'adresse 0416. Le programme se déroule alors selon la séquence normale.

Ces deux exemples illustrent le fait que chaque code d'opération différent entraîne une action particulière de l'unité de contrôle au moment du cycle d'exécution.

On peut donc résumer ainsi le rôle de l'unité de contrôle: obtenir

une instruction de la mémoire à l'adresse désignée par le compteur ordinal, augmenter d'autant la valeur de ce compteur ordinal, veiller à faire exécuter l'instruction qui vient d'être chargée dans ses registres et enfin recommencer le cycle jusqu'à ce qu'un arrêt survienne.

7.3. Instructions câblées et non câblées.

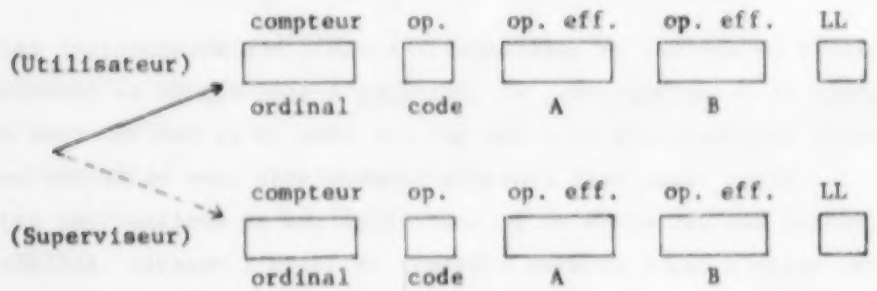
Déf: Une instruction câblée est une instruction dont le code d'opération est reconnaissable par l'unité de contrôle qui peut activer des circuits électroniques nécessaires à son exécution.

Par exemple, sur tous les ordinateurs, l'addition est câblée en ce sens que l'unité arithmétique et logique dispose des circuits électroniques capables de prendre en charge l'opération. Par contre, il arrive que des ordinateurs ne possèdent pas les circuits électroniques beaucoup plus complexes et coûteux qui exécuteraient la multiplication. On dira alors que la multiplication n'est pas câblée.

Comment donc réagit l'unité de contrôle devant un code d'opération non câblé comme la multiplication (code 52) sur le FICTIF-2? L'explication que nous donnerons ici est sommaire car des explications détaillées sont prévues dans un chapitre ultérieur. Tout d'abord, l'unité de contrôle laisse intact le contenu de ses registres pour ne pas perdre le fil du programme. En second lieu, l'unité de contrôle passe à un état particulier qui lui permet, sans déranger ses registres, d'exécuter une section de programme située dans la seconde moitié de la mémoire. Cette section de programme prend précisément en charge l'exécution de l'instruction non câblée. Dans le cas d'une multiplication, cette section de programme exécute la multiplication par une combinaison d'additions. Toutefois, même dans cet état particulier, l'unité de contrôle ne peut échapper au mode de fonctionnement que nous avons décrit plus haut. Un second ensemble de registres (compteur ordinal, op. eff. A, op. eff. B, op. code, LL) est nécessaire.

Lorsqu'une section de programme servant à exécuter une instruction non câblée est terminée, l'unité de contrôle laisse cet état particulier, appelé "mode superviseur", pour revenir à son état normal, appelé "mode utilisateur", et continuer le programme là où il avait été laissé.

L'unité de contrôle possède donc les registres suivants:



En mode normal, c'est le groupe du haut qui est utilisé. En mode particulier, le groupe du haut reste inchangé et le groupe du bas sert à faire exécuter le sous-programme approprié.

Chapitre 8.

L'unité arithmétique et logique.

Les instructions que l'unité arithmétique et logique du FICTIF-2 prend directement en charge sont l'addition, la soustraction et la comparaison. Elles sont câblées en ce sens que les circuits électroniques nécessaires à leur exécution sont physiquement présents dans cette unité.

Les instructions de multiplication et de division, par contre, ne sont pas câblées. Lorsque l'unité de contrôle détecte l'une d'elles dans un programme, l'ordinateur change d'état, c'est-à-dire, passe à l'état superviseur et exécute dans ce mode le sous-programme approprié. Les deux sous-programmes sont situés dans la deuxième moitié de la mémoire principale. Le sous-programme de multiplication utilise une combinaison d'additions et celui de la division, une combinaison de soustractions. Indirectement, l'unité arithmétique et logique contribue donc à leur exécution.

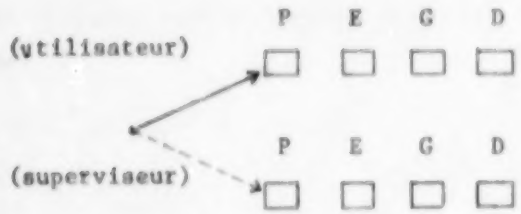
Soit dit en passant, on peut imaginer un ordinateur FICTIF-3, plus puissant que le FICTIF-2, dont les codes d'opération seraient strictement les mêmes que ceux du FICTIF-2, mais dont la multiplication et la division seraient câblées et s'exécuteraient, de ce fait, beaucoup plus rapidement. Les programmes en langage machine seraient identiques mais, pour leur exécution, l'ordinateur FICTIF-3 passerait moins souvent à l'état superviseur et aurait un rendement meilleur.

Comme l'unité de contrôle, l'unité arithmétique et logique possède ses registres particuliers. Après une instruction de comparaison, cette unité ajuste l'un des trois registres suivants à 1 et les deux autres à 0:

Les registres P (plus petit), E (égal) et G (plus grand), d'un octet chacun, ne doivent pas être confondus avec les registres d'index. Dans une instruction de saut conditionnel (code 21 à 26), l'unité de contrôle consulte le contenu de ces registres pour déterminer si le saut doit se faire ou non.

Un quatrième registre D (Dépassement), d'un octet également, sert à la fin d'une addition pour signaler un report et à la fin d'une soustraction pour signaler un emprunt. Pour plus de détails, consultez la section 6.3.

Comme l'ordinateur peut fonctionner en deux modes (utilisateur et superviseur), il y a également deux groupes correspondants de registres dans l'unité arithmétique et logique.



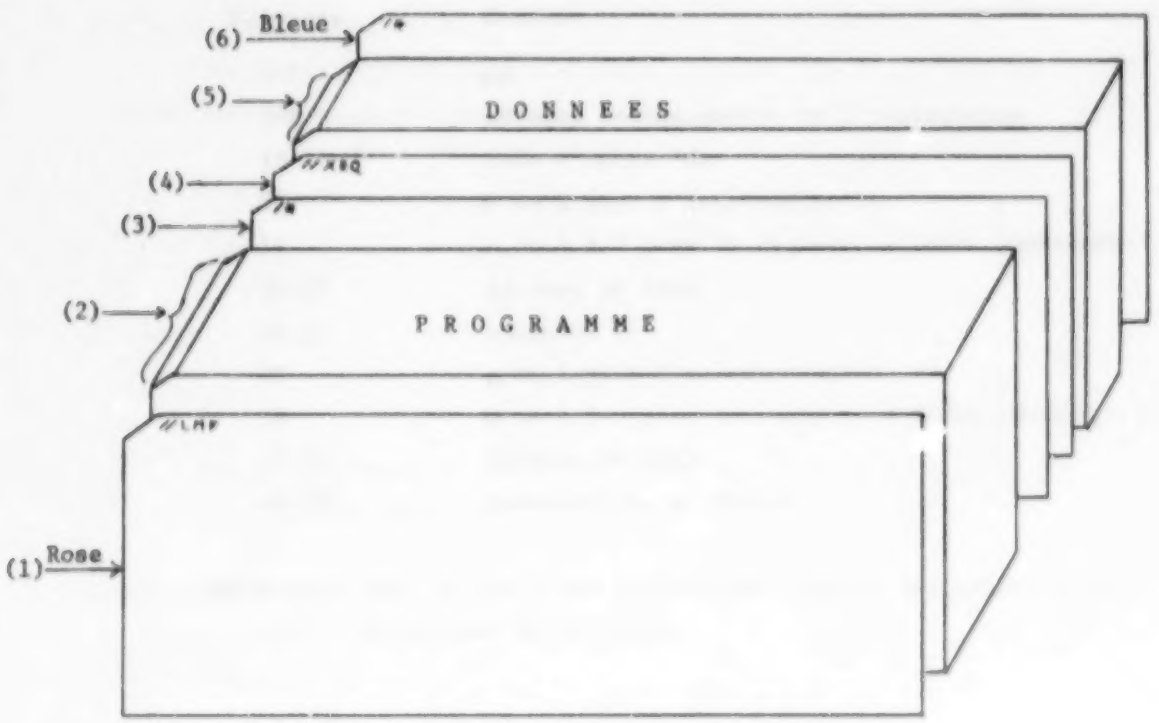
En mode normal (utilisateur), c'est le groupe du haut qui est utilisé. En mode particulier (superviseur), le groupe du haut reste inchangé et c'est celui du bas qui sert.

Chapitre 9.

Programmation en LMF (Langage Machine Fictif).

Lorsqu'il a été question d'instructions jusqu'ici, nous avons toujours admis qu'elles étaient situées en mémoire centrale. Nous allons maintenant décrire la façon d'entrer ces instructions en mémoire. Quant à la manière dont les programmes du système d'exploitation introduisent ce programme en mémoire, l'étudiant trouvera des détails au chapitre 11 réservé à l'étude du moniteur, du chargeur LMF et du compilateur (assembleur) SYMBOLIC.

La figure suivante montre comment organiser la présentation des cartes d'un programme.



(1) La carte de contrôle //LMF.

Cette carte avise le système que les cartes suivantes constituent un programme en langage machine à introduire en mémoire.

Colonnes	Contenu
1-5	//LMF
15-22	Nom de programme (un groupe non blanc de caractères).
30-49	Nom du programmeur.
55-74	Nom du professeur.

(2) Le programme.

a) Les cartes d'instructions.

Une carte est nécessaire pour chaque instruction.

Colonnes	Contenu	
1-2	␣␣	
5-8	Adresse de chargement de l'instruction.	
15-16	Code d'opération.	
18	␣ ou 1 pour l'incrémentation.	Opérande A
19	␣ ou 1 à 9 pour le registre d'index.	Opérande A
20-23	Adresse de base.	Opérande A
26-27	Longueur LL.	
29	␣ ou 1 pour l'incrémentation.	Opérande B
30	␣ ou 1 à 9 pour le registre d'index.	Opérande B
31-34	Adresse de base.	Opérande B
40-80	Commentaire, si désiré.	

Important: dans le cas d'une instruction courte, les colonnes 26 à 34 doivent être libres.

b) Les cartes de commentaires.

Des cartes ne contenant que des commentaires peuvent être placées n'importe où après la carte de contrôle.

Colonnes	Contenu
1-2	C∅
3-80	Commentaire.

c) La carte FF.

Cette carte marque la fin physique des instructions.

Colonnes	Contenu
1-2	FF
3-80	Laisser libre.

d) Les cartes de constantes.

Ces cartes permettent d'introduire des constantes en mémoire avant que le programme commence à s'exécuter.

Colonnes	Contenu
1-2	∅∅
5-8	Adresse de chargement de la constante.
15-16	Longueur de la constante (01 à 60).
21-...	La constante.

Note: des cartes de commentaires peuvent être aussi placées au travers des cartes de constantes.

(3) La carte /*.

Cette carte marque la fin du programme (instructions + constantes + commentaires).

Colonnes	Contenu
1-2	/*
3-80	Laisser libre.

(4) La carte //XEQ.

Cette carte sert à demander l'exécution du programme qui vient d'être chargé en mémoire. Des détails supplémentaires doivent être fournis sur cette carte, à savoir si des bandes doivent être montées sur l'unité 3 et/ou 4, si une image-mémoire est demandée après l'exécution, la catégorie du travail.

Colonnes	Contenu
1-5	//XEQ
11-12	S'il y a lieu, no de bande à monter sur l'unité 3.
13	P si la bande montée sur l'unité 3 doit être protégée. Sinon, laisser libre.
16-17	S'il y a lieu, no de bande à monter sur l'unité 4.
18	P si la bande montée sur l'unité 4 doit être protégée. Sinon, laisser libre.
25	Catégorie: un chiffre de 0 à 9 (0 par défaut). La catégorie contrôle le temps d'exécution alloué, le nombre de sauts de page permis et le nombre maximum de lignes à imprimer.
30-33	DUMP si une image-mémoire est désirée.
34	Si 30-33 = DUMP et que 34 = C, l'impression de l'image-mémoire sera conditionnelle: elle n'aura lieu que si le programme se termine anormalement.
35-38	Adresse terminale de l'image-mémoire. (Par défaut, 1000 est assumé.)

Note: Le programmeur ne doit jamais présumer qu'une bande est déjà montée et omettre de l'indiquer, s'il en a besoin.

(5) Les cartes de données.

Ces cartes sont lues durant l'exécution du programme, en réponse à des instructions de lecture (90). Il n'y a pour elles aucun format prédéterminé étant donné que le programmeur est pleinement responsable de définir comme il l'entend le contenu des 80 colonnes de ces cartes.

(6) La carte /*.

Il s'agit d'une carte sentinelle de fin de travail. Elle doit être bleue pour faciliter le travail de l'opérateur.

Colonnes	Contenu
1-2	/*
3-80	Laisser libre.

Remarque: Pour des motifs de clarté, toutes les colonnes non spécifiées dans tous les formats de carte précédents doivent être laissées libres.

A la page suivante, dans la partie encadrée, on trouvera un exemple d'un groupe complet de cartes respectant les règles que nous avons définies.

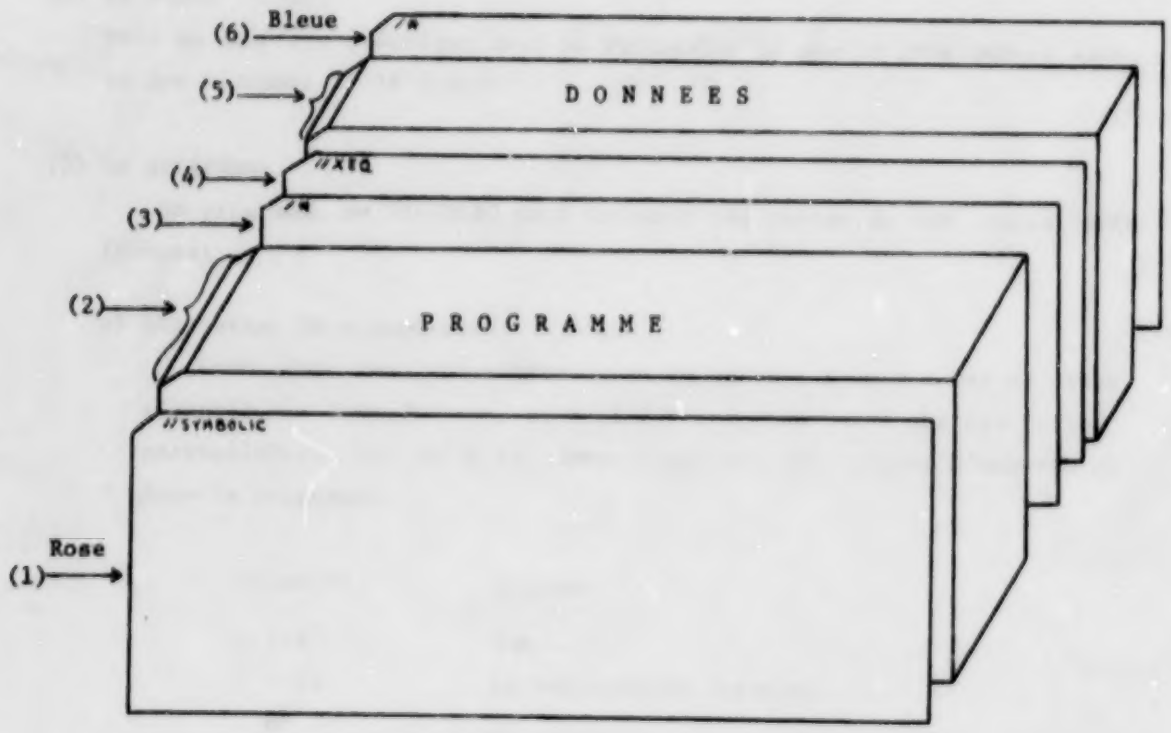
```
COLONNES
0         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

//LIF          AC102141          MARC LEBRUN                    JACQUES LEROY
     0080      54    0904  02    1100      1 DANS I
     0096      90    0001  80    1000      LIRE UNE CARTE (X)
     0112      56    1000  04    0906      COMP. X ET 5000
     0128      23    0160                  SI PLUS PETIT OU EGAL, ALLER A 0160
C       CECI EST UNE CARTE DE COMMENTAIRE
     0136      54    0911  03    1010      'NON' PRES DE X
     0152      27    0176                  SAUT POUR IMPRIMER
     0160      54    0908  03    1010      'OUI' PRES DE X
     0176      91    0002  80    1000      IMPRIMER
     0192      56    1100  02    0906      COMP. I ET 10
     0208      22    0240                  SI =, SAUT A LA FIN
     0216      54    0904  02    1100      ADD 1 A I
     0232      27    0096                  SAUT A LA LECTURE
     0240      35    0000                  ARRET
FF
     0900      14    5000011ODUINON
/*
//XEQ                    0     DUMP 0920
0200
7000
5000
6230
0235
5021
2304
1024
3402
5021
/*

0         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

Chapitre 10.

Programmation en SYMBOLIC.

Dans ce langage de programmation, il n'est plus nécessaire d'effectuer un choix des adresses et d'écrire les instructions en fonction de ce choix. Le programmeur utilise des noms de variable auxquels le compilateur assigne automatiquement des adresses. Son travail s'en trouve simplifié, étant donné qu'il peut utiliser les mêmes noms de variable que dans ses organigrammes.

Un travail se présente ici globalement de la même façon que dans le cas du LMF, sauf que la première carte contient //SYMBOLIC au lieu de //LMF.



(1) La carte de contrôle //SYMBOLIC.

Cette carte indique au système que les cartes subséquentes constituent un programme en langage SYMBOLIC à traduire en langage machine et à entrer en mémoire.

Colonnes	Contenu
1-10	//SYMBOLIC
15-22	Nom de programme (un groupe non blanc de caractères).
30-49	Nom du programmeur.
55-74	Nom du professeur.

(3), (5) et (6):

Voir au chapitre précédent.

(4) La carte //XEQ.

Voir au chapitre précédent, avec la particularité que le programmeur laisse les colonnes 35-38 libres.

(2) Le programme.

Un programme en SYMBOLIC peut contenir des cartes de six types différents:

a) Les cartes de commentaire.

Comme dans les cinq autres types de cartes, il n'y a pas de place réservée aux commentaires, ceux-ci doivent être faits sur des cartes particulières. Une carte de commentaire peut être placée n'importe où dans le programme.

Colonnes	Contenu
1-2	C ⌀
3-79	Le commentaire lui-même.
80	⌀

b) Les cartes d'index (s'il y a lieu).

Colonnes	Contenu
1-2	/*
16-18	IND
22	Le nom de l'index.
80	*

Règles: 1- Le programmeur peut définir jusqu'à 9 noms d'index, à raison d'un par carte. Le compilateur assigne IR1 au premier, IR2 au second et ainsi de suite.

2- Un nom d'index est constitué d'un seul caractère alphabétique (A ↔ Z).

3- La longueur implicite que le compilateur assigne à un index est de huit octets. Cependant, les quatre derniers seulement sont significatifs.

4- Les cartes d'index, s'il y en a, doivent précéder celles des formats c), d), e) et f).

c) Les cartes de déclaration de zones.

Colonnes	Contenu
1-2	/*
16-18	DCL
22-38 max.	Nom de zone. Possiblement, ce nom peut être suivi du nom d'une autre zone à redéfinir.
51-53	Longueur de zone, ajustée à droite. (Min: 1, Max: 999)
80	*

Règles: 1- Le nom de zone est formé d'un groupe de 1 à 6 caractères dont le premier doit être alphabétique et les autres, une combinaison de caractères numériques et/ou alphabétiques.

Ex: A
B340A
ESSENT
C180

2- Si le programmeur désire que cette zone redéfinisse une zone déjà déclarée antérieurement, c'est-à-dire qu'elles partagent la même partie de la mémoire centrale, la forme à utiliser est la suivante:

		longueur
DCL	nom-1,nom-2	---

ou encore

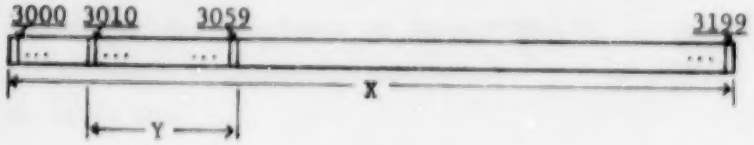
DCL	nom-1,nom-2+nnn	---

nom-1 est le nom de la zone qu'on définit et nom-2 est le nom de la zone qui est redéfinie. L'utilisation d'un déplacement nnn entre 1 et 999 permet d'établir une superposition plus à droite.

Ex:

DCL	X	200
.		
.		
.		
DCL	Y,X+10	50

Supposons que l'adresse assignée à X soit 3000, la superposition s'établira ainsi:



Y sera situé à partir de 3010.

Sur ce point, il est important de remarquer que le compilateur autorise, dans le cas d'une redéfinition, un débordement à droite de la zone redéfinissante sur la zone redéfinie. Le programmeur ne doit utiliser cette possibilité qu'avec une grande prudence.
Dans l'exemple antérieur, Y est la zone redéfinissante et X la zone redéfinie.

3- Le nom de la zone redéfinie doit être apparu antérieurement dans un DCL ou un ***.

d) Les cartes de subdivision (ou élément) de zone.

Toute carte DCL peut être suivie d'une suite de cartes de subdivision (cartes ***).

Colonnes	Contenu
1-2	**
16-18	***
22-27	* ou nom d'élément. Ces colonnes sont libres si l'espace correspondant à cette subdivision est inutilisé.
51-53	Longueur de l'élément ou de l'espace inutilisé.
80	*

Règles: 1- Le nom de l'élément est constitué d'un groupe de 1 à 6 caractères dont le premier doit être alphabétique et les autres, une combinaison de caractères alphabétiques et/ou numériques.

2- Ce nom ne doit pas être apparu antérieurement.

3- L'addition des longueurs de chaque élément ne doit pas dépasser la longueur apparaissant sur la dernière carte DCL.

Ex:

```
DCL    CARTE          80
***    PART1          10
***                   20   } total = 70 < 80.
***    PART2          40
```

Dans cet exemple, la deuxième carte *** ne porte pas de nom d'élément. L'information éventuelle contenue dans CARTE entre le onzième et le trentième caractère inclusivement ne sera donc pas utilisée.

e) Les cartes d'instruction.

Colonnes	Contenu
1-2	≠≠
3-8	Une étiquette, s'il y a lieu. Une étiquette est un nom placé devant une instruction permettant d'y faire référence, notamment pour les sauts conditionnels et inconditionnels.
16-18	Code d'opération symbolique.
22-55 max.	Opérandes.
80	≠

-L'étiquette:

Un nom d'étiquette est constitué d'un groupe de 1 à 6 caractères dont le premier doit être alphabétique et les autres, une combinaison de caractères alphabétiques et/ou numériques.

-Les codes d'opération symboliques.

Le tableau de la page suivante donne l'équivalence entre les codes d'opération en langage machine et ceux en langage symbolique.

50	ADD	Additionner	22	SEG	Saut si =
51	STR	Soustraire	23	SPE	" <
52	MUL	Multiplier	24	SPG	" >
53	DIV	Diviser	25	SDF	" ≠
54	DEP	Déplacer	26	SGE	" ≥
55	EDT	Editer	27	SAU	Saut inconditionnel
56	COM	Comparer	30	REM	Rembobiner
58	SAR	Saut avec adresse	31	PAG	Saut de page
		de retour	35	ARR	Arrêt
20	SSD	Saut si dépassement	90	LIR	Lire
21	SPP	Saut si <	91	ECR	Ecrire

-Les opérandes.

Le contenu de ces colonnes (22-55) varie selon le type d'instruction. On retrouvera possiblement un opérande symbolique A (OSA), un opérande symbolique B (OSB) et une longueur numérique (LL). Ils sont écrits dans des colonnes consécutives, sans aucun espace libre, et séparés par une virgule.

Ex: code	opérandes	sont présents:
SAU	P1	OSA
SAR	I,P1	OSA,OSB
ADD	X,Y,5	OSA,OSB,LL
LIR	1,CARTE,80	OSA,OSB,LL
ADD	A,B	OSA,OSB

Dans le dernier cas, la longueur n'a pas été donnée. Le compilateur assume 08 par défaut. Il est possible d'omettre cette longueur dans les instructions ADD, STR, MUL, DIV, DEP, EDT et COM. C'est toujours 08 qui est assumé par défaut.

De manière générale, les opérandes symboliques A (OSA) et B (OSB) peuvent obéir à plusieurs types différents. Voici les neuf (9) types possibles:

(1) ABCDEF

(2) ABCDEF+nnnn

(3) ABCDEF(I)

(4) ABCDEF(I+)

(5) ABCDEF+nnnn(I)

(6) ABCDEF+nnnn(I+)

(7) nnnn

(8) nnnn(I)

(9) nnnn(I+)

ABCDEF symbolise un nom de zone ou d'étiquette qui, dans la pratique, est composé de 1 à 6 caractères. Ce peut être aussi un nom d'index qui n'est alors composé que d'un seul caractère.

(I) veut dire que l'opérande est indexé par un registre d'index dont le nom a été défini sur une carte IND.

(I+) veut dire que l'opérande est indexé par un registre d'index et que l'incrémentation est demandée.

nnnn est une valeur numérique d'un à quatre chiffres dont la signification varie selon le contexte: dans (2), (5) et (6), nnnn représente un déplacement à ajouter à l'adresse de base de la variable représentée par ABCDEF. Dans (7), nnnn est soit un opérande immédiat (1, 2, 3 ou 4) servant à désigner un numéro de périphérique, soit une adresse absolue. Dans (8) et (9), nnnn peut être considéré comme un déplacement à ajouter à l'adresse contenue dans le registre d'index symbolisé par I.

Note: une <u>adresse absolue</u> est une adresse non symbolique, donnée directement en décimal.
Ex: <u>3000</u>.

Supposons que l'adresse de la variable X est 3000 et que J désigne le registre d'index IR2. Dans les exemples qui suivent, nous montrerons comment un opérande symbolique est traduit en

opérande d'instruction machine.

opérande symbolique	incrémentation / index / adresse	
X	00	3000
X+50	00	3050
X+50(J)	02	3050
X+50(J+)	12	3050
X(J)	02	3000
X(J+)	12	3000
1	00	0001 (opérande immédiat)
2000	00	2000 (adresse absolue)
0(J)	02	0000
5(J+)	12	0005
X+9996	00	2296 (le report est ignoré)

pour qu'un opérande soit légal, cette valeur doit être < 5000.

Voici maintenant, pour chaque catégorie d'instructions, le format général permis pour les opérandes:

Convention: une accolade $\left\{\begin{matrix}\\\\\end{matrix}\right.$ indiquera qu'un choix doit être fait en faveur d'une ligne qu'elle contient. Un crochet [ ] est utilisé dans le cas d'une option: son contenu peut être présent ou pas. Cela est possible pour la longueur LL. Si une longueur n'est pas indiquée dans une instruction où elle peut l'être, '08' est pris par défaut.

1) code — opérandes

$$\left\{\begin{matrix}\text{ADD}\\\text{STR}\\\text{MUL}\\\text{DIV}\\\text{EDT}\\\text{COM}\end{matrix}\right\} \quad \text{opérandeA,opérandeB}\left[\text{,LL}\right]$$

opérandeA, opérandeB : un des 9 types déjà cités.

$1 \leq LL \leq 16$ et = 08 par défaut.

2) code opérandes

LIR (1 / 3 / 4), opérandeB, 80

ECR (2 / 3 / 4), opérandeB, 80

- opérandeB → un des neuf types déjà nommés.
- (2 / 3 / 4) → opérande immédiat.

3) code opérandes

(SSD / SPP / SEG / SPE / SPG / SDF / SGE / SAU) opérandeA

- opérandeA → type (1), (2) ou (8), ABCDEF étant un nom d'étiquette.

4) code opérandes

REM (3 / 4)

PAG 2

- 2 → opérande immédiat.

5) ARR

- → aucun opérande.

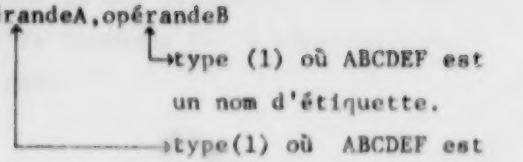
6) code opérandes

DEP opérandeA, opérandeB, LL

- opérandeB → un des 9 types déjà nommés.
- LL → $1 \leq LL \leq 80$ et = 08 par défaut.

7) code opérandes

SAR opérandeA,opérandeB

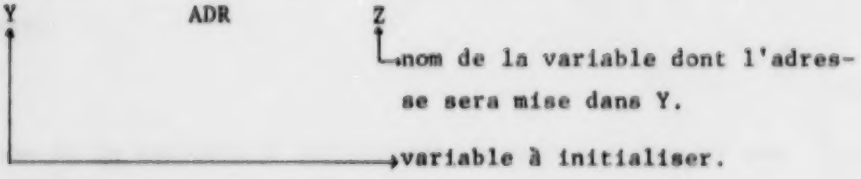


f) Les cartes d'initialisation, s'il y a lieu.

Ces cartes doivent suivre les instructions. Elles permettent d'assigner une valeur à une variable avant le début de l'exécution du programme. Quand le code de la carte est "=ØØ", cette valeur est une constante de 1 à 40 octets inclusivement. Quand ce code est "ADR", la valeur est une adresse de quatre octets. Dans ce dernier cas, il faut nommer la variable dont on veut que l'adresse soit chargée en mémoire.

Ex:

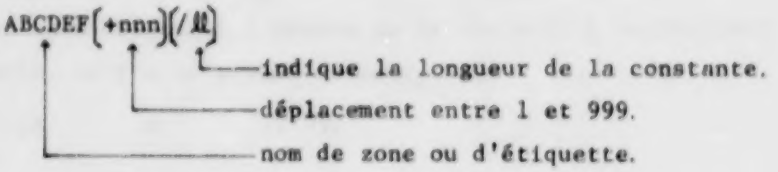
	code	valeur
X	=	450
Y	ADR	Z



Le format général de ces cartes est le suivant:

Colonnes	Contenu
1-2	ØØ
3-15 max.	Variable à initialiser.
16-18	{= / ADR}
22-61 max.	{constante de 1 à 40 octets si le code est = / nom de variable si le code est ADR.}

Soit ABCDEF le nom de la variable à initialiser. La zone 3-15 est assez longue pour contenir ce nom et des précisions sur un déplacement éventuel de même que sur la longueur de la constante qui est introduite. Le format général sera



Supposons que X ait 20 octets de long. On pourrait initialiser X de deux manières:

a) d'un seul coup:

variable	code	constante ou adresse
X	=	cte de 20 octets

b) en plus d'un coup:

X/10	=	10 octets
X+10/10	=	10 octets suivants

Cette dernière méthode est nécessaire pour initialiser au complet des champs de plus de 40 octets. Dans la première ligne sont initialisés les 10 premiers octets de X et dans la seconde, les 10 suivants.

Règles:

1- Le nom de la variable à initialiser ne doit pas être le nom d'un index.

2- Si l'initialisation d'une variable se fait d'un coup et que cette variable a 40 octets ou moins, il n'est pas nécessaire d'indiquer une longueur ℓℓ . Un déplacement, dans ce cas, n'aurait pas sa raison d'être.

3- Dans le cas d'une carte ADR, si la variable à initialiser a plus de 4 octets de long, la présence de ℓℓ = 4 est obligatoire.

4- Si la longueur de la constante excède celle, implicite ou explicite, de la zone à emplir, les octets de trop (à droite) sont ignorés. Si elle est plus courte, les blancs suivant la constante sont considérés comme en faisant partie.

5- Si une longueur explicite M est donnée, elle ne doit pas occasionner un débordement à droite de la variable à initialiser. Par exemple, si Y a 10 octets de long,

Y/15 = ...

est illégal.

Dans la partie encadrée de la page suivante, on trouvera l'image d'un ensemble de cartes (programme-source) tel que présenté par un programmeur en vue d'une compilation en SYMBOLIC et d'une exécution.

IMPORTANT: Le compilateur assume qu'un nom de variable qui n'a pas été signalé dans une carte IND, DCL ou *** mais qui apparaît dans une instruction est le nom d'une zone de 8 octets.

```
COLONNES
0        1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

//SYMBOLIC     AC1021MI      MARC LEBRUN                JACQUES LEROY
               IND    J
               DCL    TABX                          80
               DCL    CARTE                         80
               ***    Y                             4
               ***                                  76
               DCL    C9999                         4
  LECT         LIR    1,CARTE,80
               COM    Y,C9999,4
               SEG    FIN
               DEP    C0,J
  COMP         COM    Y,TABX(J),4
               SEG    IMPRES
               COM    J,C76
               SEG    LECT
               ADD    C4,J
               SAU    COMP
  IMPRES       ECR    2,CARTE,80
C   SEULES LES CARTES TROUVEES SONT IMPRIMEES.
               SAU    LECT
  FIN          ARR
  C9999        =      9999
  C0           =      00000000
  C76          =      00000076
  C4           =      00000004
  TABX/40      =      1000200030004000500060007000800090000000
  TABX+40/40   =      1200050002000333050004201230456023507542
/*
//XEQ                 0     DUMP
1000
2300
0333
9999
/*

0        1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

Chapitre 11.

Eléments de système d'exploitation.

On pourrait définir le système d'exploitation d'un ordinateur comme l'ensemble des programmes que le fabricant livre en même temps que la machine et qui sont indispensables à son fonctionnement immédiat. Par exemple, beaucoup de systèmes d'exploitation contiennent un compilateur COBOL permettant la traduction en langage machine de programmes écrits dans ce langage de programmation universel. On exclut cependant de cette catégorie les programmes d'application (paye, gestion de stock, etc.) qui pourraient également être achetés du fabricant.

Les quatre programmes du système d'exploitation de l'ordinateur FICTIF-2 dont nous parlerons dans ce chapitre sont:

a) le moniteur d'enchaînement (programme réglant l'enchaînement des travaux sur l'ordinateur);

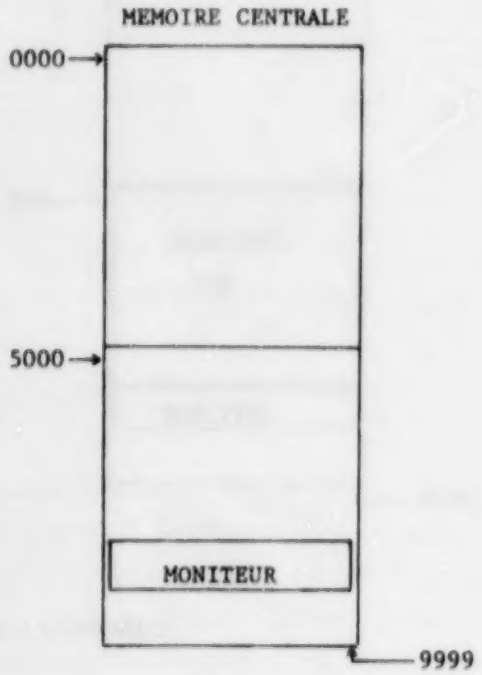
b) le chargeur LMF (programme permettant d'entrer en mémoire un programme écrit en LMF);

c) le compilateur (ou assembleur) SYMBOLIC (programme permettant de traduire et de charger en mémoire un programme écrit en SYMBOLIC);

d) le générateur d'image-mémoire (programme ayant pour fonction d'imprimer une image-mémoire).

11.1. Le moniteur d'enchaînement.

Le moniteur, comme les autres programmes du système d'exploitation, réside en permanence sur une bande magnétique et peut être apporté en mémoire centrale en temps opportun. Au moment de la mise en marche quotidienne de l'ordinateur, le moniteur est chargé en mémoire à l'endroit indiqué dans la figure suivante:

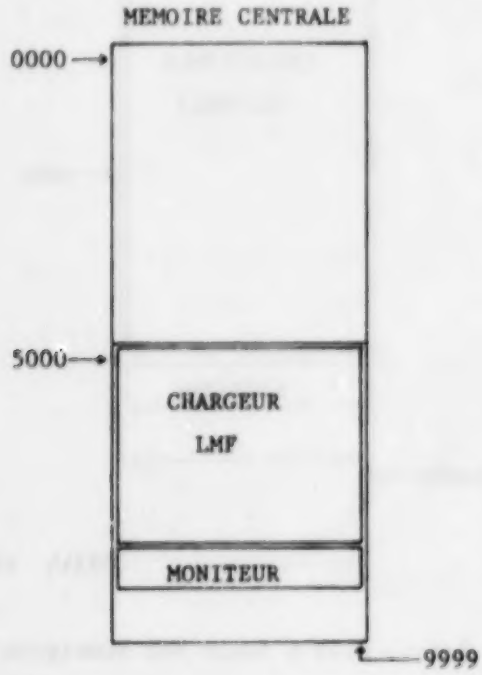


Il n'est sans doute pas inutile de signaler que tous les programmes du système d'exploitation résidant sur bandes sont déjà en langage machine et directement exécutables par l'ordinateur.

Le moniteur est un programme dont le rôle est d'analyser les cartes de contrôle et de prendre les décisions appropriées. Ces cartes de contrôle sont:

a) la carte //LMF.

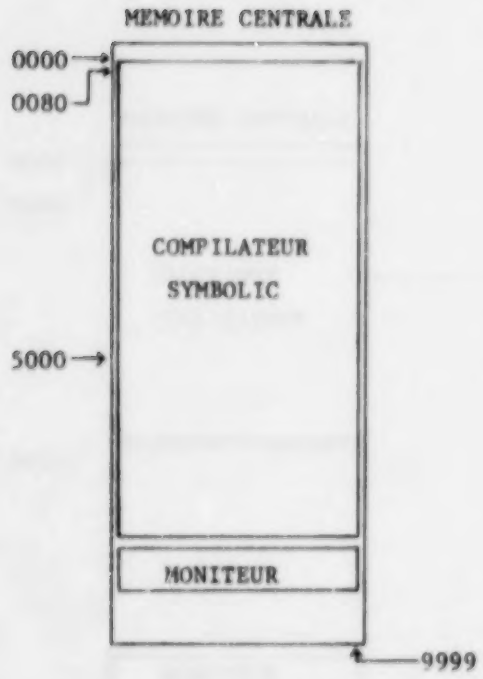
Si elle est valide, le MONITEUR exécute le chargement, à partir d'une bande magnétique du système, du programme "CHARGEUR LMF" à compter de l'adresse 5000. Lorsque ce chargement est terminé, le MONITEUR passe le contrôle au CHARGEUR LMF.



b) la carte //SYMBOLIC.

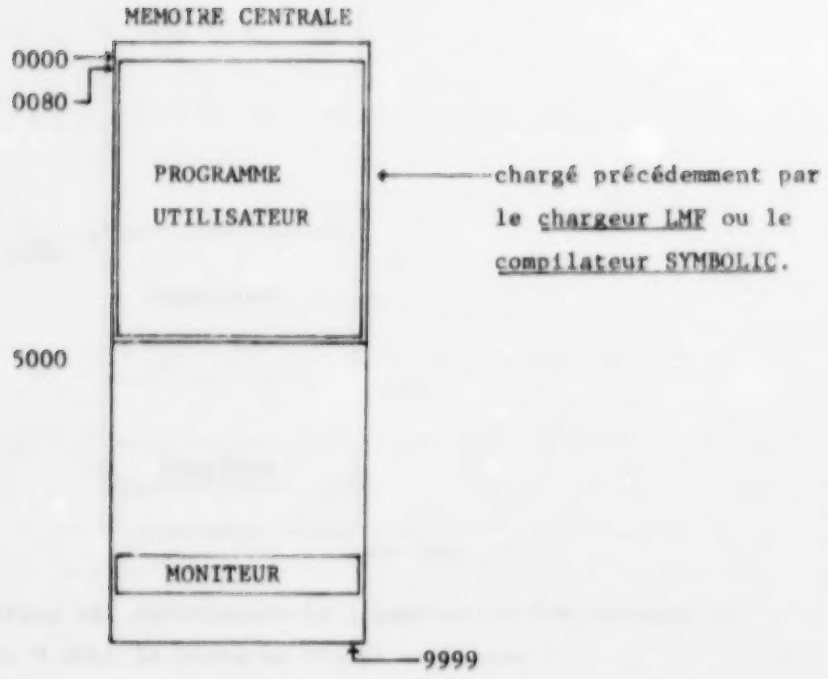
Si elle est valide, le MONITEUR exécute le chargement, à partir d'une bande magnétique du système, du programme "COMPILATEUR SYMBOLIC" à compter de l'adresse 0080. Lorsque le chargement est terminé, le MONITEUR passe le contrôle au COMPILATEUR SYMBOLIC.

A cause de sa dimension plus grande, ce programme occupe une bonne partie de la mémoire, comme le montre la figure suivante:

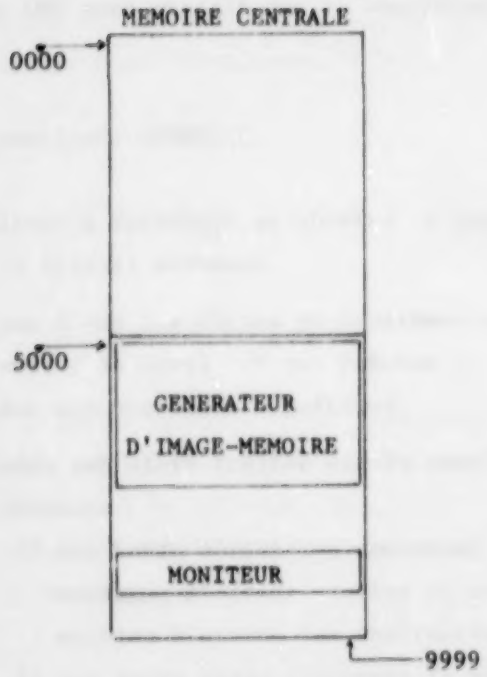


c) la carte //XEQ.

Si le programme qui vient d'être traité par le Chargeur LMF ou le Compilateur SYMBOLIC est valide et si cette carte //XEQ est correcte en tout point, le moniteur passe le contrôle au programme (de l'utilisateur-étudiant) situé à partir de l'adresse 0080. Le contrôle reviendra au moniteur lorsque l'instruction d'arrêt (35000000) sera exécutée.



Le moniteur utilise de plus une information (le mot DUMP) sur cette carte //XEQ pour déterminer si une image-mémoire doit être produite à la fin de l'exécution du programme-utilisateur. Il retient aussi l'adresse terminale de cette image-mémoire. Au moment où cette dernière doit être produite, le moniteur exécute le chargement, à partir d'une bande magnétique du système, du programme "GENERATEUR D'IMAGE-MEMOIRE" à compter de l'adresse 5000. Lorsque le chargement est terminé, le MONITEUR passe le contrôle à ce programme en lui communiquant l'adresse terminale de l'image-mémoire.



Le moniteur est responsable de l'impression des messages de la catégorie M dont la liste se trouve en annexe 1.

11.2. Le chargeur LMF.

Après que le moniteur a introduit en mémoire le chargeur LMF, ce dernier effectue le travail suivant:

a) il lit une à une les cartes du programme-source jusqu'à temps de rencontrer la carte /* qui précède la carte //XEQ.

b) il les imprime au fur et à mesure, produisant ainsi un "listing".

c) il les valide une à une et imprime, s'il y a lieu, un des messages de la catégorie L dont la liste est en annexe 1. L'impression d'un message se fait immédiatement sous la ligne de la carte en erreur.

d) tant qu'aucune erreur n'est rencontrée, il effectue le chargement en mémoire des intructions et des constantes.

Lorsque ce travail est terminé, le chargeur LMF retourne le contrôle au MONITEUR qui analyse la demande d'exécution (carte //XEQ) et y donne suite

selon que le chargeur LMF aura signalé que le chargement s'est fait sans erreur ou pas.

11.3. Le compilateur (assembleur) SYMBOLIC.

Après que le moniteur a introduit en mémoire le compilateur SYMBOLIC, ce dernier effectue le travail suivant:

a) il lit une à une les cartes du programme-source jusqu'à temps de rencontrer la carte /* qui précède la carte //XEQ et les emmagasine sur une bande magnétique.

b) cette bande est alors traitée par le compilateur jusqu'à ce que soient obtenues

1) une bande définitive contenant le "listing" complet, messages d'erreur inclus et traduction en langage machine à droite des instructions symboliques;

2) une autre bande contenant le programme intégral en langage machine, y compris les initialisations de constantes.

c) il procède à l'impression de la bande contenant le "listing" complet. Les étoiles ajoutées en colonne 80 permettent de retracer aisément le point à partir duquel le programme est impropre à être chargé en mémoire en vue d'une exécution.

d) Si aucune erreur n'a été détectée durant la compilation, il charge en mémoire à partir de l'adresse 0080 le programme intégral se trouvant sur bande. La première partie du compilateur située de 0080 à 4999 est alors détruite. Les conséquences sont nulles car la section effectuant ce travail-ci est située plus loin en mémoire.

Lorsque tout ce travail est terminé, le compilateur SYMBOLIC (ou plus précisement ce qui en reste) retourne le contrôle au MONITEUR qui analyse la demande d'exécution (carte //XEQ) et y donne suite selon que le compilateur

SYMBOLIC aura signalé que la traduction s'est faite sans erreur ou non. Le compilateur communique également au moniteur l'adresse maximale utilisée durant la compilation en vue de déterminer où s'arrêtera l'impression éventuelle de l'image-mémoire.

Le compilateur est responsable des messages de type X dont la liste est en annexe 1.

11.4. Le générateur d'image-mémoire.

Quand le moniteur a chargé en mémoire le générateur d'image-mémoire à partir de l'adresse 5000, il lui passe le contrôle et l'adresse terminale de l'image-mémoire à imprimer. Ce dernier effectue alors l'impression de ce rapport dont nous avons parlé au chapitre 2 (section 2.4.). Le même exemple, non sectionné, se trouve à la page suivante.

Note: Le lecteur sera peut-être étonné du fait que ses programmes ne peuvent pas dépasser l'adresse 4999 alors que le compilateur va de 0080 à une adresse bien supérieure à 5000. Il en va de même pour le chargeur et pour le générateur d'image-mémoire. La réponse réside dans le fait que le moniteur peut modifier cette valeur limite au besoin et la fixer à 4999 avant de passer le contrôle à un programme de l'utilisateur, à 7499 par exemple avant de passer le contrôle au compilateur, et ainsi de suite. En somme, il s'agit de comprendre que cette valeur limite supérieure est modifiable, mais uniquement par le système.

```
0000* COKLMNOP0000 0076 R                  0032*
0064*  ABCDEFGH   *<(+  9000000180000464   0096* 9100000280000624 5400046480000544
0128* 5400078408000792 5400080008000008   0160* 5601046404010468 2300023254010464
0192* 5400080854010468 0401046454000808   0224* 0401046856000792 0800000822000280
0256* 5000081608000008 2700016056000792   0288* 0800080022000328 5100081608000792
0320* 2700014454600765 7900070454000600   0352* 0800000854010464 0400073854010544
0384* 0400072491000002 8000070456000008   0416* 0800082422000456 5000081608000008
0448* 2700036035000000 0000000201010230   0480* 0232050212342023 2032212022222222
0512* 2222230233333333 3520356542325452   0544* 1234023020320101 0502423254523520
0576* 0000023200622222 2120222223023565   0608* 2222202333333333
0640*   NO  TRIES            TRIES         0672*
0704*                        3333          0736*   5452
0768*                   0000007200000000   0800* 000000002032     0000000400000076
```

PARTIE II.

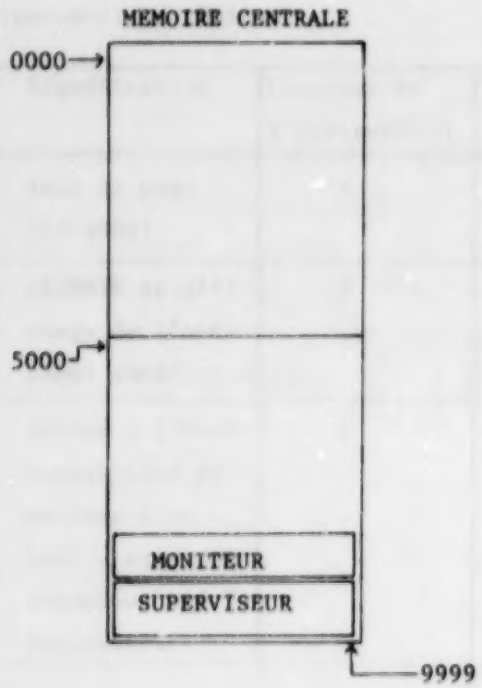
Chapitre 12.

Le superviseur.

12.1. Généralités.

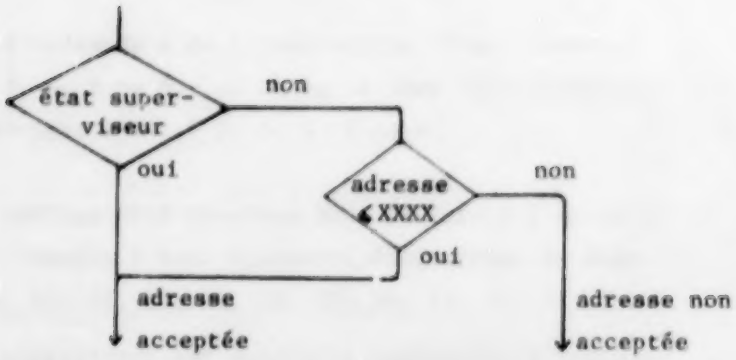
De tous les programmes du système d'exploitation, le superviseur est le plus fondamental car, tant qu'il n'est pas en mémoire, le fonctionnement normal de l'ordinateur n'est pas assuré. Sans le nommer, nous y avons fait allusion au sujet de la multiplication et de la division non câblées.

Une fois en mémoire à l'endroit indiqué dans la figure suivante, ce programme a la particularité que tous les sous-programmes qui le composent s'exécutent en mode superviseur, cet état particulier dont nous avons parlé au chapitre 7.



Deux faits importants caractérisent l'état superviseur:

a) dans cet état, l'accès complet de la mémoire est autorisé. On pourra concevoir ainsi le fonctionnement du dispositif de protection de la mémoire:



Cette valeur XXXX, nous l'avons déjà dit, est modifiable et ajustée à 4999 avant que le contrôle soit donné à un programme-utilisateur.

b) A l'état superviseur, l'unité de contrôle reconnaît des codes d'opération dit <u>privilégiés</u> qui ne peuvent être exécutés que dans cet état. Ils correspondent à des instructions câblées.

Code d'opération	Signification	Longueur de l'instruction	Longueur des opérandes
19	Saut de page (A = 0002)	8	-
29	ATTENTE et affichage de l'opérande immédiat A	8	-
39	Retour à l'état superviseur en mettant l'op. imm. A dans COS (compteur ord./ superviseur)	8	-
49	Rembobiner (op. imm. A = unité)	8	-
59	Lire	16	80
69	Ecrire	16	80

Dans ce dernier tableau, l'opérande A de l'instruction 59 est immédiat et il désigne l'unité 1, 3, 4, 5 ou 6. Il en va de même dans l'instruction 69 où l'opérande A désigne l'unité 2, 3, 4, 5 ou 6.

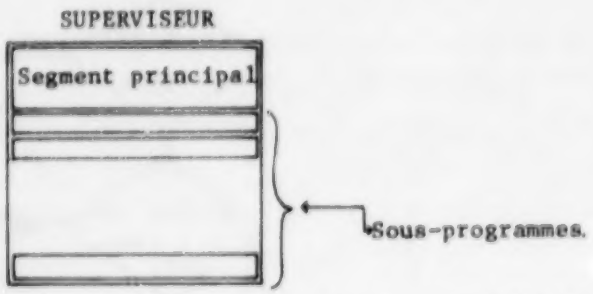
Toutes les instructions câblées déjà décrites dans la Partie I et qu'on peut retracer au tableau de l'annexe 2 sont également disponibles en état superviseur. Ce sont: 50, 51, 54, 56, 20, 21, 22, 23, 24, 25, 26, 27.

En résumé, le programme superviseur est constitué uniquement d'instructions câblées et lui seul peut contenir des instructions privilégiées.

12.2. Constitution du superviseur.

Comme il a été dit, le superviseur est constitué d'un groupe de sous-programmes. Chaque sous-programme a un rôle particulier. L'un prend en charge l'exécution de la multiplication, un autre de la division; il y en a pour contrôler les opérations d'entrée-sortie, etc. En somme, toutes les instructions non-câblées mentionnées dans la Partie I sont prises en charge par un sous-programme du superviseur. D'autres sous-programmes sont utilisés dans des situations d'erreur. Le superviseur est responsable des messages de la catégorie S dont la liste se trouve en annexe 1. Un coup d'oeil sur la nature de ces erreurs donne une idée des contrôles exercés par ce dernier.

Il faut cependant remarquer la présence d'un segment principal dont le rôle est de déterminer quel sous-programme doit être exécuté.



Si tous les sous-programmes du superviseur ne peuvent résider en mémoire, faute de place, on peut aisément supposer que le segment principal peut les appeler d'une bande du système au moment voulu.

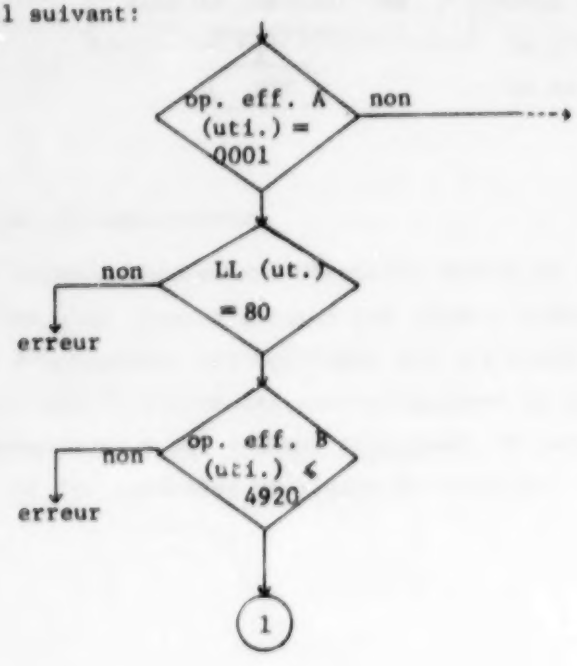
12.3. Fonctionnement.

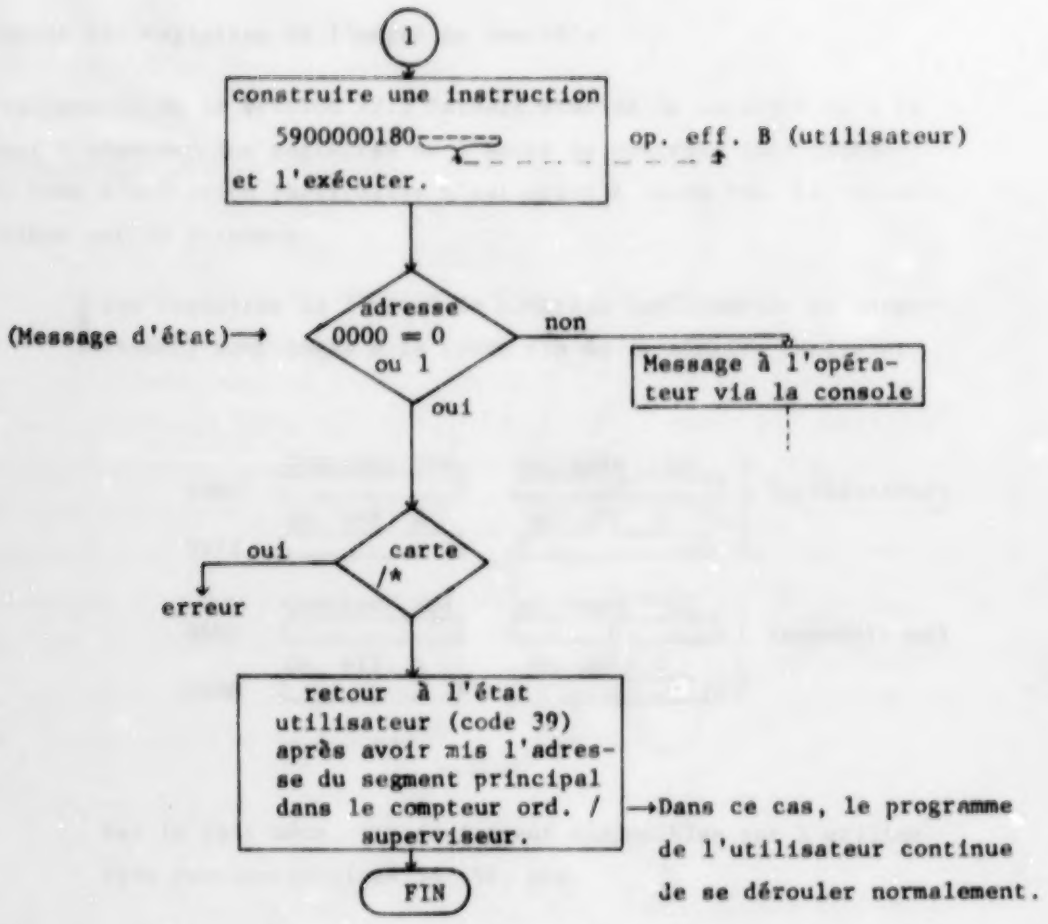
Pour faire comprendre le fonctionnement du superviseur, nous allons nous restreindre à un cas particulier, laissant le soin à l'imagination de chacun d'échafauder, à partir des éléments donnés, la solution à apporter aux autres cas.

Supposons que le programme de l'utilisateur contienne l'instruction suivante à l'adresse 2000 et qu'elle soit à exécuter immédiatement:

9000000180003000

L'unité de contrôle charge cette instruction dans ses registres (utilisateur) et augmente le compteur ordinal (utilisateur) à 2016. Reconnaissant un code d'opération non câblée, l'unité de contrôle change d'état et passe à l'état superviseur. Si l'on admet que le compteur ordinal (superviseur) contient précisément l'adresse du segment principal du superviseur dont nous venons de parler, ce segment principal analysera le code d'opération qui se trouve dans les registres de l'unité de contrôle, côté utilisateur. Identifiant le code 90, il passera le contrôle au sous-programme concerné qui effectuera le travail suivant:





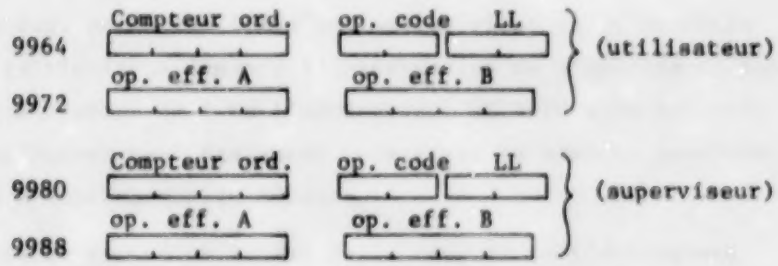
12.4. Réflexions au sujet du superviseur.

Le superviseur donne une puissance virtuelle accrue au matériel. Il est même possible d'aller plus loin, d'ajouter des codes d'opération non câblés pour effectuer les 4 opérations arithmétiques sur des zones comportant un signe + ou -. Il suffirait d'ajouter des sous-programmes au superviseur et d'apporter quelques changements à son segment principal. De cette manière, le superviseur extensionne les capacités physiques du matériel.

12.5. Emplacement des registres de l'unité de contrôle.

L'organigramme de la section 12.3 faisait état de la capacité qu'a le superviseur d'examiner les registres de l'unité de contrôle (utilisateur). Or, aucun code d'opération particulier n'est prévu à cette fin. La réponse à ce problème est la suivante:

> les registres de l'unité de contrôle (utilisateur et superviseur) sont logés à la toute fin de la mémoire centrale:



Par le fait même, ils deviennent disponibles par l'utilisation des instructions 54, 56, etc.

Note: XXXX, la valeur limite utilisée par le dispositif de protection de la mémoire est logée immédiatement après ces registres, c'est-à-dire aux adresses 9996-9999.

Chapitre 13.

La console (ou pupitre de commande).

13.1. Aspects généraux.

La console est le lieu d'interaction entre l'ordinateur et l'opérateur. D'une part, c'est par son intermédiaire que l'opérateur met en marche l'ordinateur, qu'il commande le chargement du superviseur et du moniteur en mémoire (par l'intermédiaire de l'amorce), qu'il met l'ordinateur en attente, qu'il peut entrer manuellement certaines informations en mémoire et même faire démarrer l'ordinateur en mode superviseur à une adresse quelconque. D'autre part, c'est par la console que l'ordinateur transmet certaines informations à l'opérateur, notamment lorsqu'une erreur vient de se produire ou qu'une condition particulière demande l'intervention de l'opérateur: les octets correspondant à l'opérande A de l'instruction ATTENTE sont affichés sur des voyants. L'opérateur peut également interroger la mémoire pour connaître le contenu de n'importe quelle adresse.

A la page suivante se trouve un schéma de la console de l'ordinateur FICTIF-2. Définissons d'abord les symboles utilisés.

: Bouton-poussoir.

: Afficheur d'un caractère décimal par un procédé au néon, comme sur les mini-calculateurs de poche.

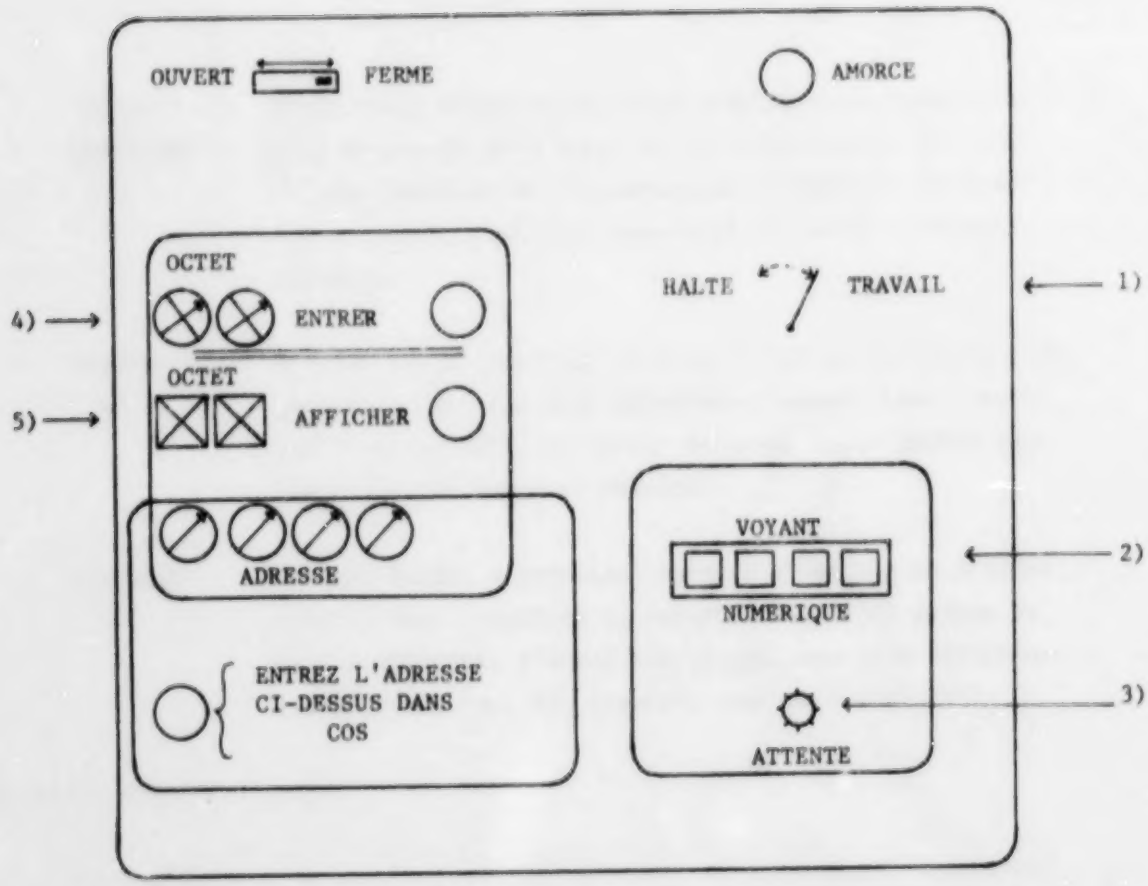
: Sélecteur décimal.

: Sélecteur hexadécimal.

: Afficheur d'un caractère hexadécimal (même procédé).

: Lampe-témoin.

COS : COMPTEUR ORDINAL (état superviseur).



AMORCE : L'opérateur, en appuyant sur ce bouton, fait lire le premier enregistrement physique de la bande système sur l'unité 5 et emmagasiner son contenu en mémoire centrale à partir de l'adresse 5000. Cette opération s'effectue en état HALTE . La mise au TRAVAIL se fera en mode superviseur.

HALTE : Toutes les opérations sur la console doivent être faites en
TRAVAIL état de HALTE. La mise au TRAVAIL implique un démarrage dans
(1) le mode où l'ordinateur était avant l'arrêt, sauf si une entrée vient d'être faite sur la console dans COS, auquel cas la reprise se fait en mode superviseur.
La mise en HALTE interrompt le signal sonore occasionné par l'instruction ATTENTE (29).

VOYANT NUMERIQUE (2) : L'exécution d'une instruction ATTENTE occasionne un signal sonore et l'allumage de la lampe-témoin (3). L'opérande immédiat de l'instruction ATTENTE est affiché sur ce voyant numérique constitué de quatre afficheurs décimaux.

ENTRER (4) : En état HALTE, une fois choisie l'adresse où entrer l'octet défini par les deux sélecteurs hexadécimaux (ex: 41 pour l'octet "A"), il suffit de presser sur ENTRER pour que cet octet entre en mémoire.

AFFICHER (5) : En état HALTE, l'opérateur choisit l'adresse de l'octet dont il veut connaître le contenu. Lorsqu'il presse le bouton AFFICHER, l'octet est visible sur deux afficheurs hexadécimaux (ex: 41 apparaît pour la lettre "A").

13.2. Précisons sur l'amorce.

Pour démarrer l'ordinateur, en plus d'ouvrir l'interrupteur principal, l'opérateur monte une bande particulière sur l'unité 5. L'ordinateur est alors en état de HALTE. Après avoir pressé le bouton AMORCE, il entre l'adresse 5000 dans COS (compteur ordinal / superviseur) et effectue la mise au TRAVAIL. L'amorce lue sur le premier enregistrement physique de la bande est en fait un programme en langage machine qui s'exécute maintenant en mode superviseur et qui fait entrer en mémoire d'autres enregistrements. Le travail de chargement s'effectue automatiquement jusqu'à ce que le moniteur et le superviseur soient en mémoire à leur place.

13.3. Exemple d'interaction entre le superviseur, la console et l'opérateur.

Supposons que dans l'exemple donné à la section 12.3, le message d'état analysé par le superviseur ait été 3 (lecteur de cartes vide). Celui-ci exécute alors l'instruction suivante

Cette instruction met l'ordinateur en attente, occasionne l'émission d'un signal sonore et affiche 3450 sur le voyant numérique de la console. L'opérateur met l'ordinateur en HALTE, ce qui arrête le signal sonore. Il consulte alors un catalogue de messages et y voit que le message 3450 signifie que le lecteur de cartes est vide. Il corrige donc la situation et fait ensuite la mise au TRAVAIL. A ce moment, l'instruction qui suit 29003450 en mémoire est exécutée et le programme se poursuit. Le superviseur pourra alors reprendre l'opération qui avait échouée.

Chapitre 14.

Contrôle interne de l'ordinateur.

On exige d'abord d'un ordinateur qu'il exécute correctement des programmes dépourvus d'erreurs. Il est cependant tout aussi important qu'il puisse détecter les anomalies et y réagir convenablement. Bien sûr, l'ordinateur pourrait les signaler à l'opérateur, par l'intermédiaire de voyants sur la console, et laisser ce dernier analyser la situation et prendre les décisions qui s'imposent; souvent il s'agira d'indiquer le type d'erreur, d'abandonner le programme en cours et de passer au travail suivant. Cette "délégation de pouvoir" serait coûteuse en temps à cause de la relative lenteur de l'homme par rapport à l'ordinateur. Les risques d'erreur aussi seraient élevés. On a donc cherché à doter la machine des moyens de se dégager seule des situations anormales. On a pu, par extension, appliquer les mêmes techniques à des situations qui ne correspondaient pas à des anomalies: ici, par exemple, on verra comment une horloge interne peut interrompre le programme en cours pour permettre un contrôle sur sa durée d'exécution.

Ce chapitre contient des explications sur la notion de trappe et sur l'usage qu'on en fait dans l'ordinateur FICTIF-2.

Déf: Une trappe est une instruction située à un endroit précis de la mémoire et que l'ordinateur est forcé d'exécuter en mode superviseur, suite à une condition particulière. Cette instruction est un saut inconditionnel à un sous-programme du superviseur.

14.1. Trappe PE (erreur de programme).

Les conditions qui conduisent l'ordinateur à prendre cette trappe sont les suivantes:

a) une adresse dépasse la valeur maximale (XXXX) permise par le dispositif de protection de la mémoire;

b) le compteur ordinal n'est pas divisible par 8.

Pour plus de précision, on pourra se reporter à l'organigramme de la section 7.2 qui explique le fonctionnement du cycle d'instruction. Nous avions dit que les sorties latérales de cet organigramme correspondaient à des situa-

tions où une procédure spéciale était appliquée. Il s'agissait, bien sûr, de la prise de cette trappe. Toutefois, l'ordinateur ne peut pas détecter les erreurs logiques d'un programme qui ne peuvent être trouvées que par une analyse détaillée des résultats. Si le programme est soumis à un ensemble suffisant de tests, les erreurs de logique apparaîtront inévitablement dans ceux-ci.

En mémoire la trappe PE se présente ainsi:

2700---- ← adresse d'un sous-programme du superviseur.

Le sous-programme a pour rôle d'indiquer l'erreur par un message, de remettre l'adresse du segment principal du superviseur dans le COS (compteur ordinal/ superviseur). Le contrôle sera alors passé au moniteur qui fera lire des cartes jusqu'à trouver une carte de contrôle.

Note: l'adresse du segment principal doit normalement être dans le COS quand un programme s'exécute en mode utilisateur, car la rencontre d'un code d'opération non câblé amène le passage au mode superviseur et la valeur du compteur ordinal ne peut pas être quelconque.

14.2. Trappe ME (erreur de mémoire).

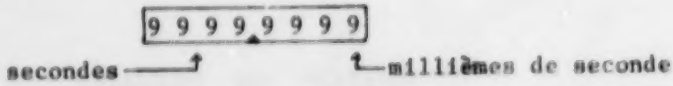
A l'intérieur de l'ordinateur, les échanges d'octets entre la mémoire et d'autres parties sont soumis à des contrôles permettant de détecter si des bits sont perdus en cours de route. Cette condition est exceptionnelle et ne survient que très rarement. La trappe ME est constituée comme la précédente:

2700---- ← adresse d'un sous-programme du superviseur.

Le sous-programme du superviseur peut être construit de façon à reprendre l'instruction manquée. Les chances que la même erreur se produise une deuxième fois sont limitées, à moins d'une défaillance permanente du matériel. Dans ce cas, cette anomalie sera détectée car le sous-programme n'autorisera qu'un nombre limité d'essais, après quoi un message devra être émis. Il y aura à ce moment interruption complète des activités et un technicien devra être appelé à réparer le matériel.

14.3. Trappe HI (horloge interne).

L'horloge interne du FICTIF-2 possède un registre qui est incrémenté automatiquement à tous les millièmes de seconde.



A intervalle régulier, par exemple à tous les centièmes de seconde, cette horloge génère une interruption de programme. Aussitôt que l'intruction en cours est terminée, le saut est fait, par l'intermédiaire de la trappe HI, à un sous-programme du superviseur qui teste si le temps alloué au programme est terminé. Le temps d'exécution d'un programme est calculé à partir de deux lectures du temps, au début et la fin, et par la soustraction de ces deux valeurs. Pour être accessible au superviseur sans code d'opération particulier, le registre de l'horloge est situé en mémoire centrale. Les positions des trois trappes et du registre de l'horloge interne sont d'ailleurs les suivantes:

9932	2700____	trappe PE
9940	2700____	trappe ME
9948	2700____	trappe HI
9956	________	registre HI.

Si le temps alloué au programme n'est pas dépassé, il y a retour au mode utilisateur et reprise du programme à l'endroit où il avait été interrompu.

ANNEXE 1

CATALOGUE DES MESSAGES

ET ERREURS.

Messages du MONITEUR.

Note: Le suffixe 'E' indique qu'à cause de cette erreur, l'exécution n'aura pas lieu. Le suffixe 'M' accompagne une erreur qui n'est normalement pas fatale.

M01-E : Sur une carte //LMF ou //SYMBOLIC, le nom de programme n'est pas présent.
Si une exécution a été demandée, une erreur 'M02-E' suivra.

M02-E : Carte //XEQ en désordre. Une exécution doit suivre directement une compilation.
Ce message peut suivre le 'M01-E'.

M03-E : Carte de contrôle de type inexistant.
Ex: //SIMBOLIC

M04-M : Pour un programme considéré exécutable, le programmeur a omis de demander l'exécution.
Vérifiez si l'ordre des cartes est correct et si //XEQ est bien écrit.

M05-M : Programme non exécutable à cause d'erreurs dans le programme-source.
Ce message n'est pas indiqué si l'exécution n'est pas demandée.

M06-M : Carte //XEQ en erreur. Vérifiez le répertoire des bandes car un numéro de bande spécifié pour l'unité 3 ou 4 est inexistant, ou encore c'est le même numéro qui est donné pour les deux.

M07-M : D'après le répertoire des bandes, la bande demandée sur l'unité 3 doit être protégée. Un 'P' est assumé.

M08-M : D'après le répertoire des bandes, la bande demandée sur l'unité 4 doit être protégée. Un 'P' est assumé.

M09-M : Sur la carte //XEQ, la catégorie spécifiée est invalide. La catégorie 0 (zéro) est assumée.

M10-M : Sur la carte //XEQ, pour un programme en LMF, une image-mémoire ou "dump" est demandée avec une adresse terminale invalide. L'adresse 1000 est assumée.

M11-E : Erreur du système d'exploitation. Avisez un responsable.

Messages du programme de chargement des travaux en LMF.

LO1-E : La première carte suivant la carte //LMF doit être une instruction dont l'adresse de chargement est 0080.

LO2-E a,b,c,...

Note: a,b,c,... sont des chiffres de 1 à 8 représentant un type d'erreur dans l'instruction.

1 : Adresse de chargement invalide car elle est soit non numérique, soit numérique mais supérieure à 4800, soit numérique mais pas divisible par 8.

2 : Code d'opération invalide. Dans ce cas, aucune validation ne se fait sur le reste de l'instruction.

3 : Opérande A (incluant l'incrémentation et l'index) non numérique.

Note: un ≠ pour l'incrémentation et l'index est traduit par 0 (zéro) et est donc valide.

4 : L'opérande A n'est pas situé dans les limites permises pour cette instruction.

5 : Pour une instruction courte, la deuxième partie doit être laissée libre.

6 : Même chose qu'en 3 sauf qu'il s'agit de l'opérande B et, en plus, de la longueur LL.

7 : La longueur LL n'est pas dans les limites permises pour cette instruction.

8 : L'opérande B n'est pas situé dans les limites permises pour cette instruction.

L03-E : Pour une carte de constante, la longueur doit être un nombre entre 01 et 60.

L04-E : Pour une carte de constante, l'adresse de chargement n'est pas numérique ou encore l'adresse plus la longueur de la constante dépasse 5000.

L05-E : Dans la suite des instructions, la carte /* a été rencontrée avant la fin des instructions.

Note: La fin physique des instructions est marquée par une carte ayant FF en colonnes 1 et 2.

L06-E : Dans la suite des instructions, la carte est ignorée car les deux premières colonnes ne sont ni ¢¢, ni /*, ni //, ni FF, ni C¢.

L07-E : Dans la suite des instructions, une carte // est rencontrée avant la fin physique des instructions, marquée par une carte FF. Dans ce cas, le chargement est abandonné et le retour se fait au moniteur.

S'il s'agit d'une carte //XEQ, un message d'erreur M02-E suivra la réimpression de cette carte sur une nouvelle page.

S'il s'agit d'une carte //LMF ou //SYMBOLIC, la carte sera réimprimée au début d'une nouvelle page et le moniteur la traitera comme marquant le début d'un nouveau travail.

S'il s'agit d'une carte // de type inconnu, le message M03-E suivra la réimpression de la carte au début d'une nouvelle page.

L0[illegible] : Dans la suite des constantes, la carte //XEQ apparaît avant qu'une carte /* ne soit rencontrée.

L09-E : Dans la suite des constantes, la carte est ignorée car les deux premières colonnes ne sont ni ¢¢, ni /*, ni //, ni C¢.

L10-E : Dans la suite des constantes, une carte // (différente de //XEQ) est rencontrée. Le travail est abandonné et un retour se fait au moniteur La carte sera imprimée en haut d'une nouvelle page.

Messages du Superviseur.

Note: Tous les messages suivants surviennent durant l'exécution du programme et ont pour conséquence l'arrêt de cette exécution. Au moment de l'impression du message, le superviseur imprime aussi le contenu des registres standards de l'unité de contrôle; à noter que le compteur ordinal indique l'adresse de l'instruction suivante.

S01 : L'unité de contrôle signale que le compteur ordinal n'est pas divisible par 8.

S02 : L'unité de contrôle signale que le compteur ordinal n'a pas pour valeur une adresse comprise entre 0080 et 4984 inclusivement.

S03-A : En obtenant les 8 premiers octets de l'instruction en mémoire, l'unité de contrôle reconnaît que l'un de ceux-ci n'est pas numérique.

S03-B : En obtenant les 8 derniers octets d'une instruction longue, l'unité de contrôle reconnaît que l'un de ceux-ci n'est pas numérique.

S04 : L'unité de contrôle ne reconnaît pas le code d'opération de l'instruction en cours.

S05 : L'unité de contrôle prévoit qu'au cours de l'exécution de la présente instruction, l'opérande A dépassera l'adresse maximale permise par le dispositif de protection-mémoire (4999).

S06 : L'unité de contrôle prévoit qu'au cours de l'exécution de la présente instruction, l'opérande B dépassera l'adresse maximale permise par le dispositif de protection-mémoire (4999).

S07 : L'unité de contrôle reconnaît l'instruction "saut de page" mais l'opérande A n'est pas 0002.

S08 : Le programme tente de dépasser le nombre de sauts de page autorisé dans la catégorie choisie.

S09 : La longueur des opérandes pour une instruction d'addition, de soustraction, de multiplication, de division, de comparaison, de déplacement ou d'édition est invalide.

S10 : Un ordre de rembobinage est donné pour une unité autre que 3 ou 4 (opérande A = 0003 ou 0004) ou encore l'unité désignée est valide mais aucune bande n'y est montée.

S11 : Le programme demande une lecture sur une unité autre que 1, 3 ou 4 (opérande A = 0001, 0003 ou 0004).

S12 : La carte qui vient d'être lue dans le lecteur de cartes est /*.

S12-A : La carte qui vient d'être lue dans le lecteur de cartes est une carte de contrôle //. Le travail est abandonné immédiatement et aucune image-mémoire ("dump") ne sera donnée. De plus, ni le temps ni le contenu des registres standards de l'unité de contrôle ne seront donnés.

S13 : Le programme demande de lire ou d'écrire sur l'unité 3 et aucune bande n'y est montée.

S14 : En essayant de lire ou d'écrire un enregistrement sur l'unité 3, la marque de fin de bande MDF a été détectée.

S15 : Le programme demande de lire ou d'écrire sur l'unité 4 et aucune bande n'y est montée.

S16 : En essayant de lire ou d'écrire un enregistrement sur l'unité 4, la marque de fin de bande MDF a été détectée.

S17 : Le programme demande une écriture sur une unité autre que 2, 3 ou 4 (opérande A = 0002, 0003 ou 0004).

S18 : Le programme tente de dépasser le nombre permis de lignes à imprimer dans la catégorie choisie.

S19 : L'unité 3 refuse d'écrire sur une bande munie de son dispositif de protection.

S20 : L'unité 4 refuse d'écrire sur une bande munie de son dispositif de protection.

S21 : Le superviseur met un terme à l'exécution du programme car le temps alloué pour la catégorie choisie est dépassé.

S22 : Dans une multiplication, un des opérandes n'est pas numérique.

S23 : Dans une division, un des opérandes n'est pas numérique ou encore une division par 0 (zéro) est demandée.

S24 : Le champ émetteur dans une opération d'édition doit être numérique mais ne l'est pas ici.

S25 : Le masque d'édition ne doit pas contenir d'autres caractères que
. , 9 Z $

S26 : Sur l'unité 3, le programmeur tente de changer l'usage d'une bande (source ou destination) ailleurs qu'au début de cette bande.

S27 : Sur l'unité 4, le programmeur tente de changer l'usage d'une bande (source ou destination) ailleurs qu'au début de cette bande.

S28 : Dans l'instruction "saut avec adresse de retour" (58), le premier opérande ne désigne pas un des registres d'index (0001 à 0009).

Messages du compilateur SYMBOLIC.

X01 : Cette carte est ignorée car les colonnes 1 et 2 ne contiennent ni ␣␣, ni /*, ni // ni C␣.

X02 : Une carte /* ou // est mal située. La compilation s'arrête immédiatement. Vérifiez l'ordre de vos cartes et l'exactitude des cartes de contrôle.

X03 : Le nom d'un index ne doit comporter qu'une lettre alphabétique.

X04 : Neuf (9) noms d'index seulement sont permis.

X05 : Le même nom d'index est défini plus d'une fois.

X06 : Dans un énoncé DCL, le nom de zone à définir dépasse 6 caractères.

X07 : Dans un énoncé DCL, le nom de la zone redéfinie dépasse 6 caractères.

X08 : Le déplacement accompagnant le nom de la zone redéfinie a plus de 3 caractères.

X09 : Dans un énoncé DCL, il n'y a aucun opérande.

X10 : Le nom de la zone à définir est invalide. Il doit être constitué d'au plus 6 caractères dont le premier doit nécessairement être alphabétique et dont les autres doivent être numériques ou alphabétiques. (Ex: A12B est un nom valide.)

X11 : Le format de l'opérande dans un énoncé DCL est invalide.

X12 : Le nom de la variable redéfinie n'est pas valide. Voyez le commentaire X10.

X13 : Dans un énoncé DCL, le déplacement accompagnant le nom de la zone redéfinie est incorrect.

X14 : Dans un énoncé DCL, le nom de la variable redéfinie n'est pas trouvé ou correspond à un nom d'index, ce qui est invalide.

X15 : Dans un énoncé DCL, la longueur de zone, telle que définie, est incorrecte.

X16 : Le nom de la zone qui vient d'être définie sera ignorée pour le reste de la compilation car la "table des symboles" est pleine. Un maximum de 200 symboles est accepté.

X17 : La compilation s'arrête immédiatement car il y a plus de 500 cartes (à partir du premier DCL jusqu'à la dernière instruction du programme, en excluant les cartes de constante).

X18 : Des caractères sont rencontrés dans une partie normalement vide de la zone des opérandes: après un premier ␣, aucun caractère ne doit apparaître.

X19 : La présente carte est ignorée parce que

a) il ne s'agit ni de DCL ni de ***;

b) il s'agit de *** mais la carte DCL précédente est invalide, ce qui rend invalides tous les *** qui lui sont assujettis.

X20 : Dans un énoncé ***, la longueur de zone, telle que définie, est incorrecte.

X21 : Les longueurs additionnées sur les cartes *** dépasse la longueur spécifiée sur la dernière carte DCL.

X22 : Sur une carte ***, le nom de zone est invalide (voir X10) ou il a déjà été déclaré antérieurement.

X23 : Dans une instruction, le code d'opération n'est pas reconnu.

X24 : La compilation s'arrête immédiatement car aucun code d'opération valide n'a été trouvé avant les cartes de constante (=xxx ou ADR).

X25-A : Dans une instruction, le nom d'étiquette est invalide (voir X10) ou ce nom existe déjà.

B : Parce que la table des symboles est pleine, ce nom d'étiquette sera ignoré pour le reste de la compilation.

C : L'opérande doit commencer par une lettre ou par un chiffre.

D : Déplacement incorrect: un + doit être suivi d'un nombre de 1 à 4 chiffres.

E : Après un nom d'index qui ne comporte qu'un caractère alphabétique,) ou +) est attendu mais pas trouvé.

F : Un nom d'index est attendu après "(" mais pas reconnu comme tel. Un nom d'index ne doit comporter qu'un seul caractère alphabétique. Il est possible que la carte définissant l'index soit en erreur.

G : Dans un opérande, le nom de variable dépasse 6 caractères.

H : L'opérande se termine par un caractère incorrect. La fin d'un opérande est marquée par ≠ ou , selon qu'il s'agit d'un opérande terminal ou pas.

I : Des caractères sont rencontrés dans une partie normalement vide de la zone des opérandes.

J : Le numéro de périphérique est invalide pour le type d'instruction utilisée.

K : Le chargement des instructions en mémoire ne peut plus avoir lieu à partir de ce point car la longueur du programme dépasse les capacités de la mémoire.

Note: lorsqu'une instruction ne peut être chargée en mémoire pour une raison au l'autre, un * suit sa traduction en langage-machine.

L : Le format de l'opérande n'est pas accepté. Vérifiez les formats permis pour ce type d'instruction.

M : Dans un saut conditionnel ou inconditionnel, le nom d'étiquette n'est pas reconnu comme tel. L'énoncé où cette étiquette se trouve contient probablement une erreur. Il est possible que cette étiquette n'existe pas.

N : Une adresse traduite en décimal est considérée trop élevée dans cette instruction.

O : Dans l'instruction SAR (Saut avec adresse de retour),
a) le premier opérande n'est pas un nom d'index accepté ou
b) le deuxième opérande n'est pas un nom d'étiquette accepté.

P : Un nom de zone non déclaré explicitement est impossible à entrer dans la table des symboles car celle-ci est déjà pleine ou encore le nom n'a pas une forme valide.

Q : La longueur spécifiée est incorrecte.

X26 : Avertissement: une carte //XEQ doit être précédée d'une carte /*.

X27 : Dans les constantes à entrer, le code n'est pas =~~p/p~~ ou ADR.

X28 : La longueur totale du programme, instructions et données, dépasse 4999. Pour être exécutable, le programme doit être raccourci.

X29 : Dans une instruction de chargement de constante, le nom de la variable est invalide ou inexistant.

X30 : Dans une carte de constante, le format de la zone de l'étiquette est inaccepté.

X31 : Dans une carte de constante, le chargement n'a pas lieu car un dépassement de mémoire est prévu.

X32 : La longueur demandée dans une instruction de chargement est invalide.
Pour ADR, elle doit être de 4.
Pour =xxx, elle doit être comprise entre 1 et 40 inclusivement.
Note: Lorsque la longueur n'est pas spécifiée dans la zone de l'étiquette, la longueur définie explicitement ou implicitement (8) est prise par défaut.

X33 : Dans la zone de l'étiquette, le déplacement est en erreur. Un + doit être suivi d'un nombre de 1 à quatre (4) chiffres.

X34 : Dans une instruction de chargement ADR, l'opérande ne contient pas un nom de variable connu.

ANNEXE 2

TABLEAU DES CODES
D'OPERATION.

Code d'opération	Code symbolique	Signification	Longueur de l'inst.	Longueur des opérandes	Câblé ou non câblé
50	ADD	A + B → B	16	01-16	C
51	STR	B - A → B	"	"	C
52	MUL	A * B → B	"	"	N
53	DIV	B ÷ A → B	"	"	N
54	DEP	A → B	"	01-80	C
55	EDT	Editer A → B	"	01-16	N
56	COM	Comp. A et B	"	"	C
58	SAR	Saut ↔ Retour	"	-	N
20	SSD	Saut si Dépas.	8	-	C
21	SPP	Saut si <	"	-	C
22	SEG	" =	"	-	C
23	SPE	" ≤	"	-	C
24	SPG	" >	"	-	C
25	SDF	" ≠	"	-	C
26	SGE	" ≥	"	-	C
27	SAU	Saut incond.	"	-	C
30	REM	Rembobiner	8	-	N
31	PAG	Saut de page	"	-	N
35	ARR	Arrêt	"	-	N
90	LIR	Lire	16	80	N
91	ECR	Ecrire	16	80	N